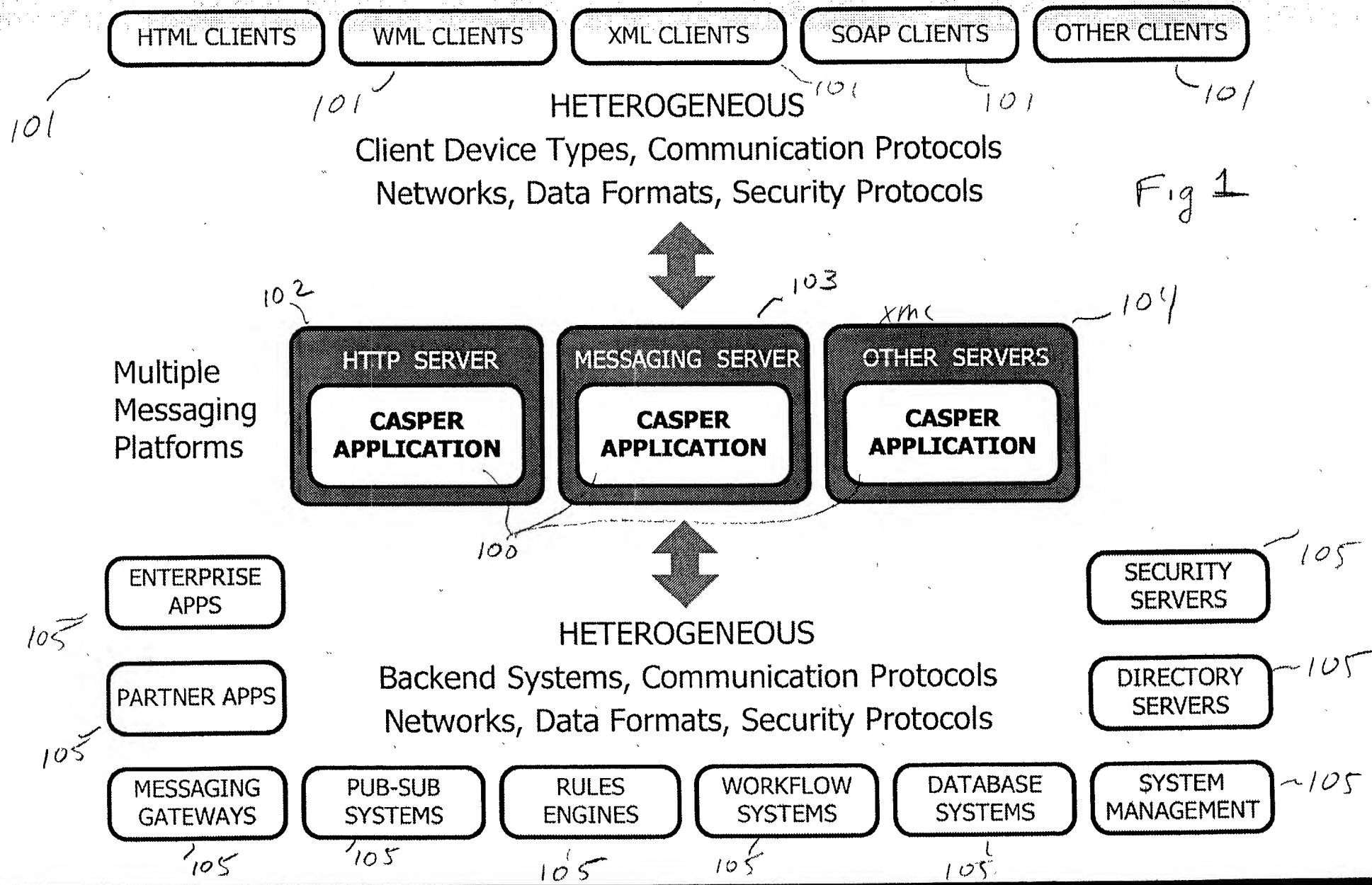
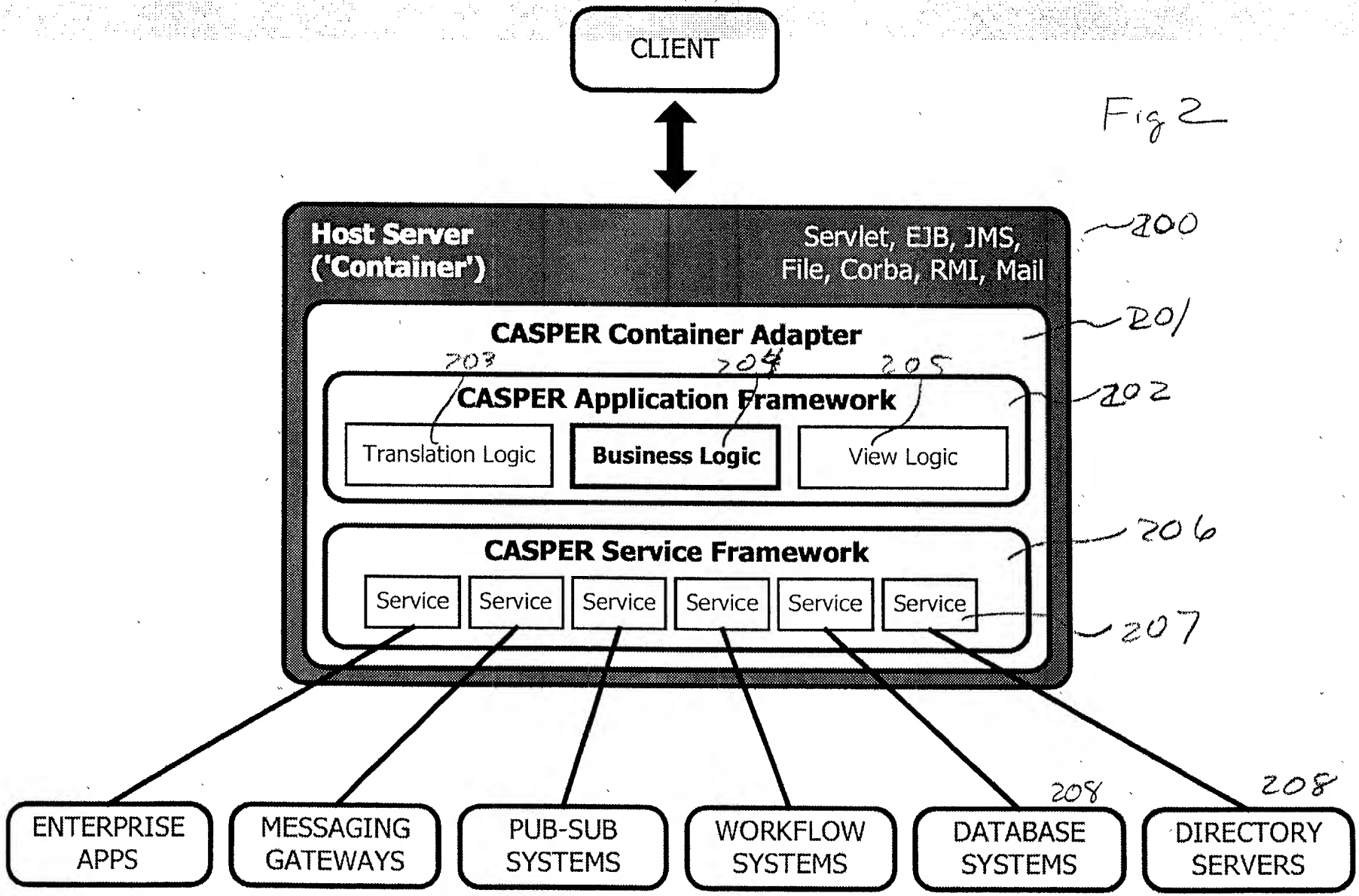
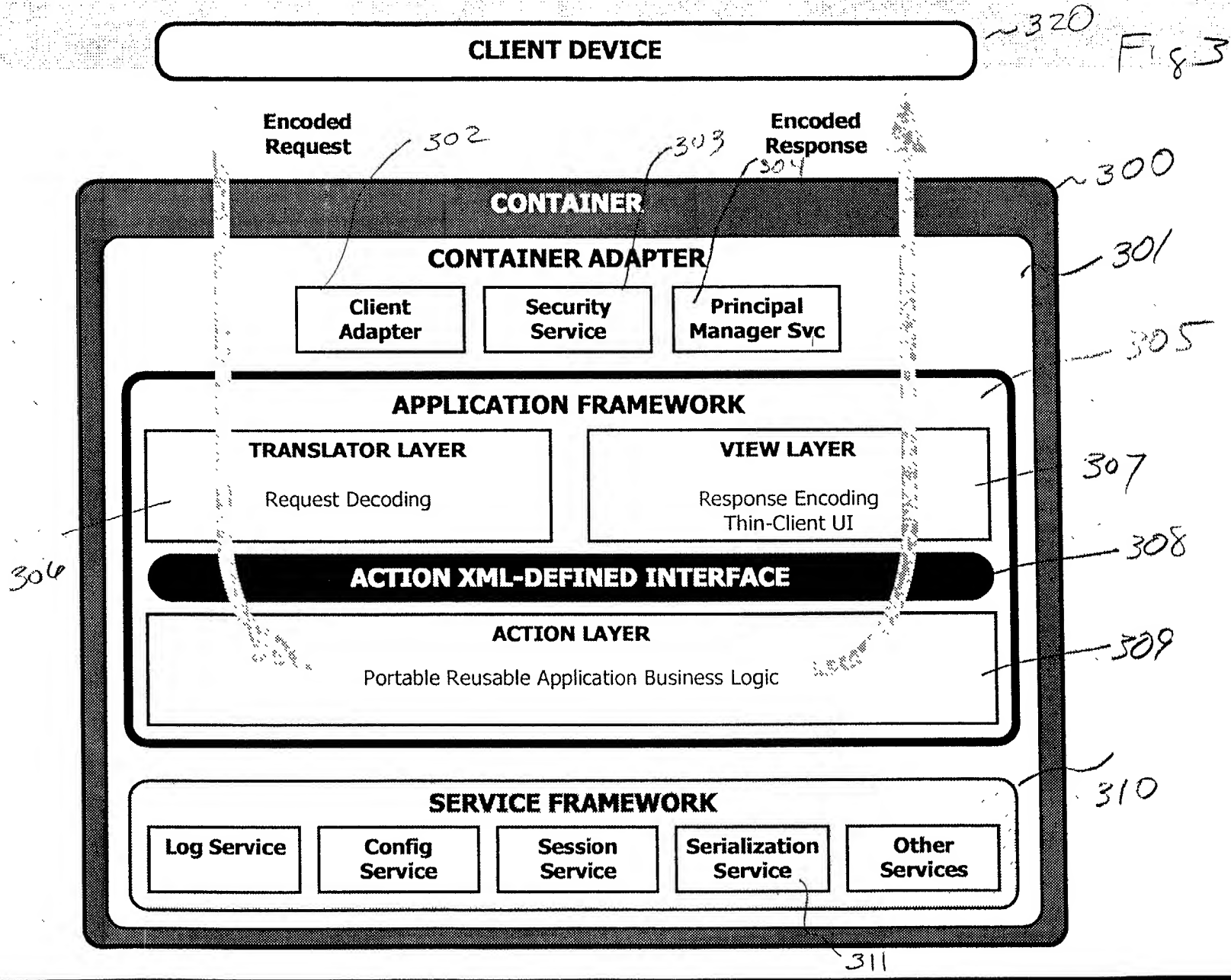
Heterogeneous Distributed Environment



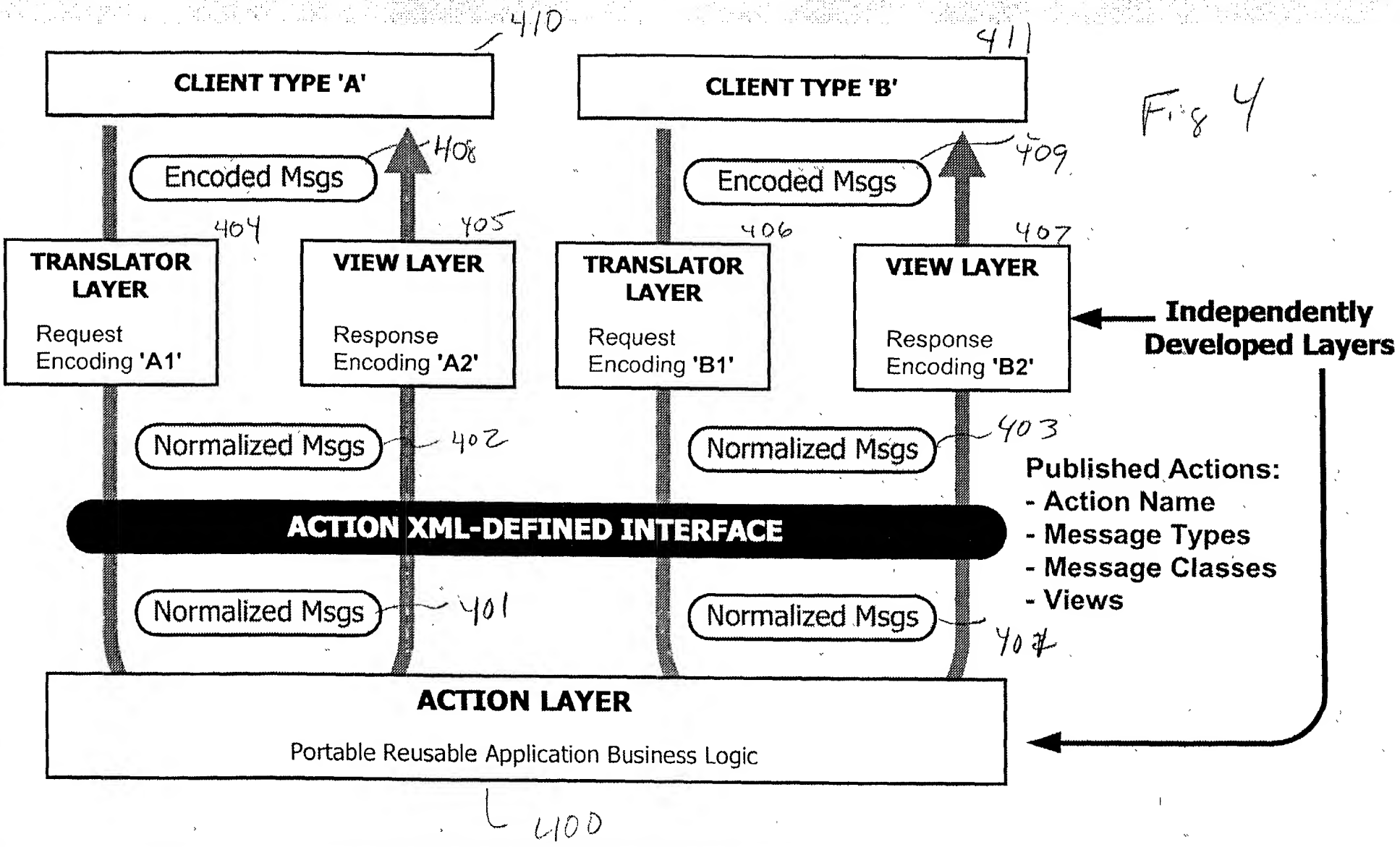
Container and Framework Overview



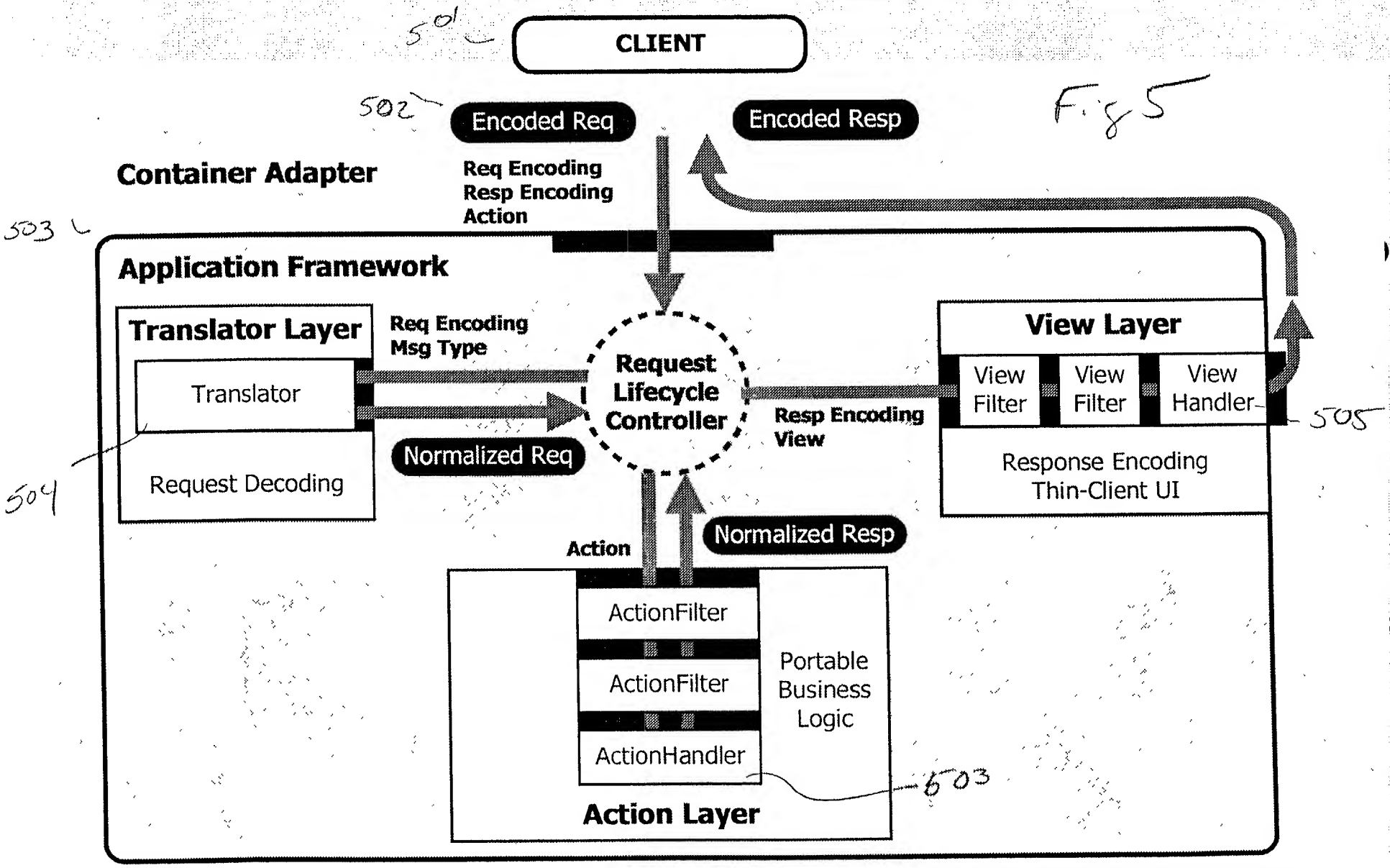
Application Framework Overview



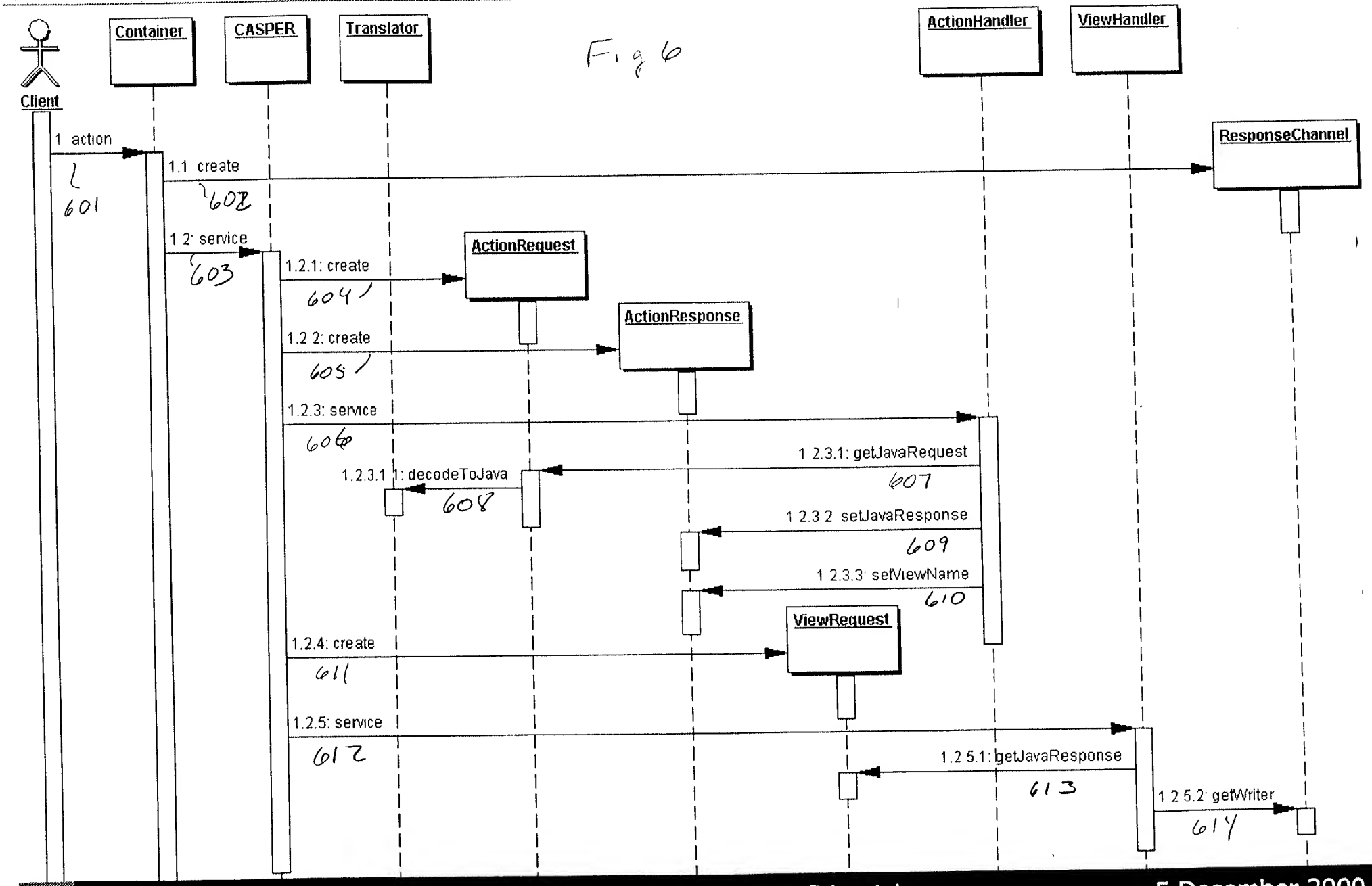
Published XML-Defined Action Interface



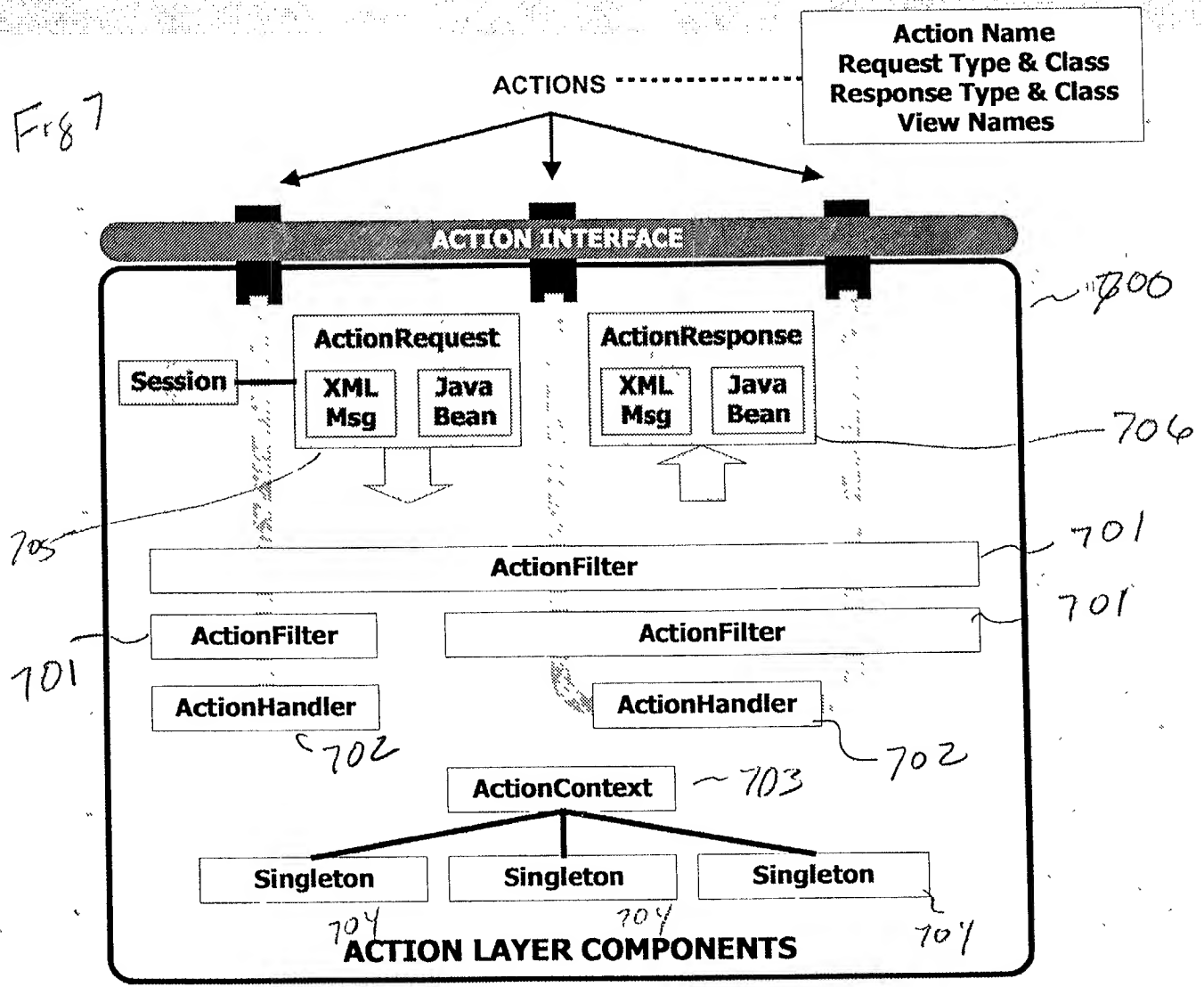
Application Request Lifecycle Overview



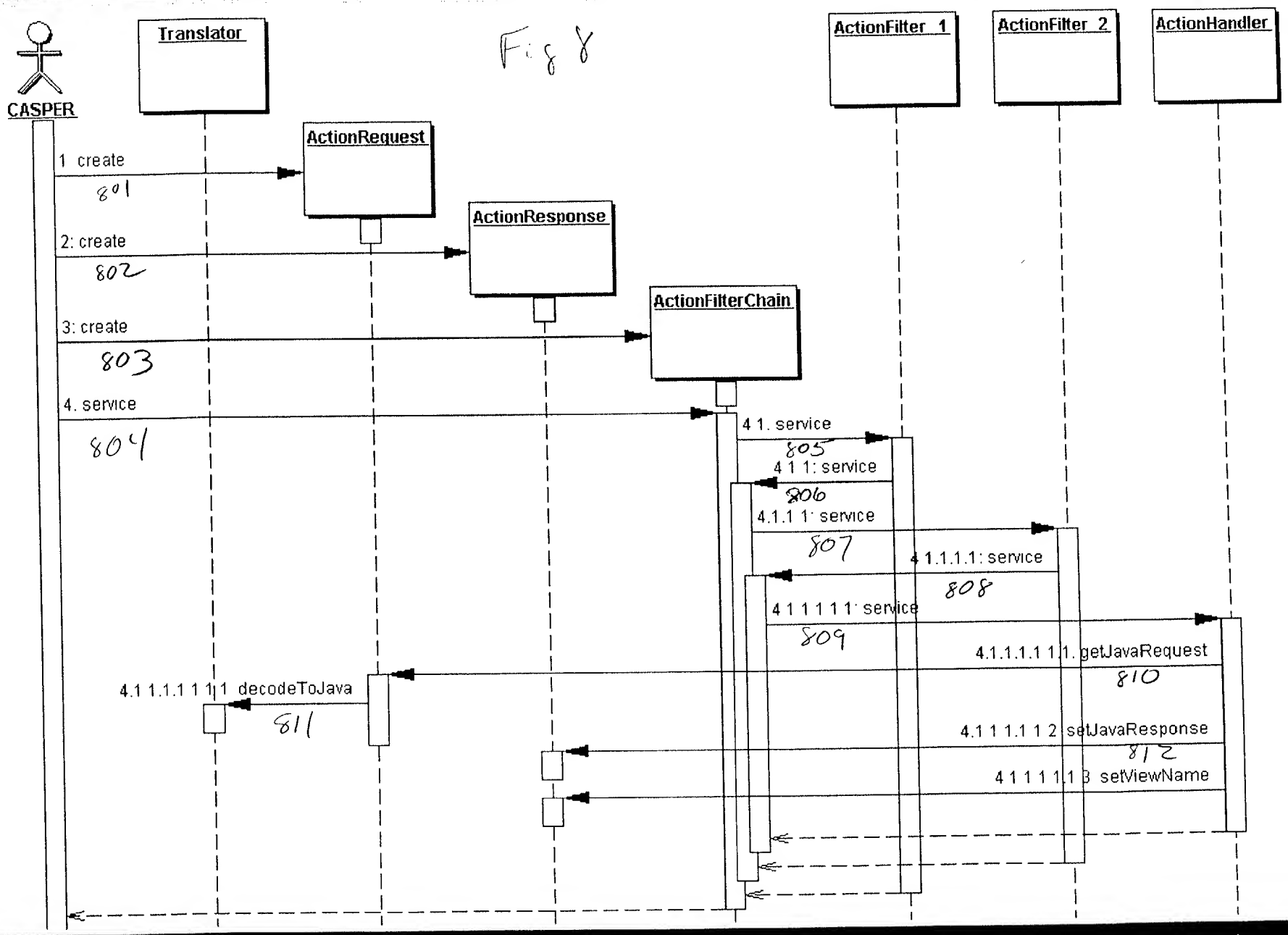
Application Request Sequence Overview



Action Layer Components



Action Layer Request Sequence Overview



Action Layer Dynamic Dispatch

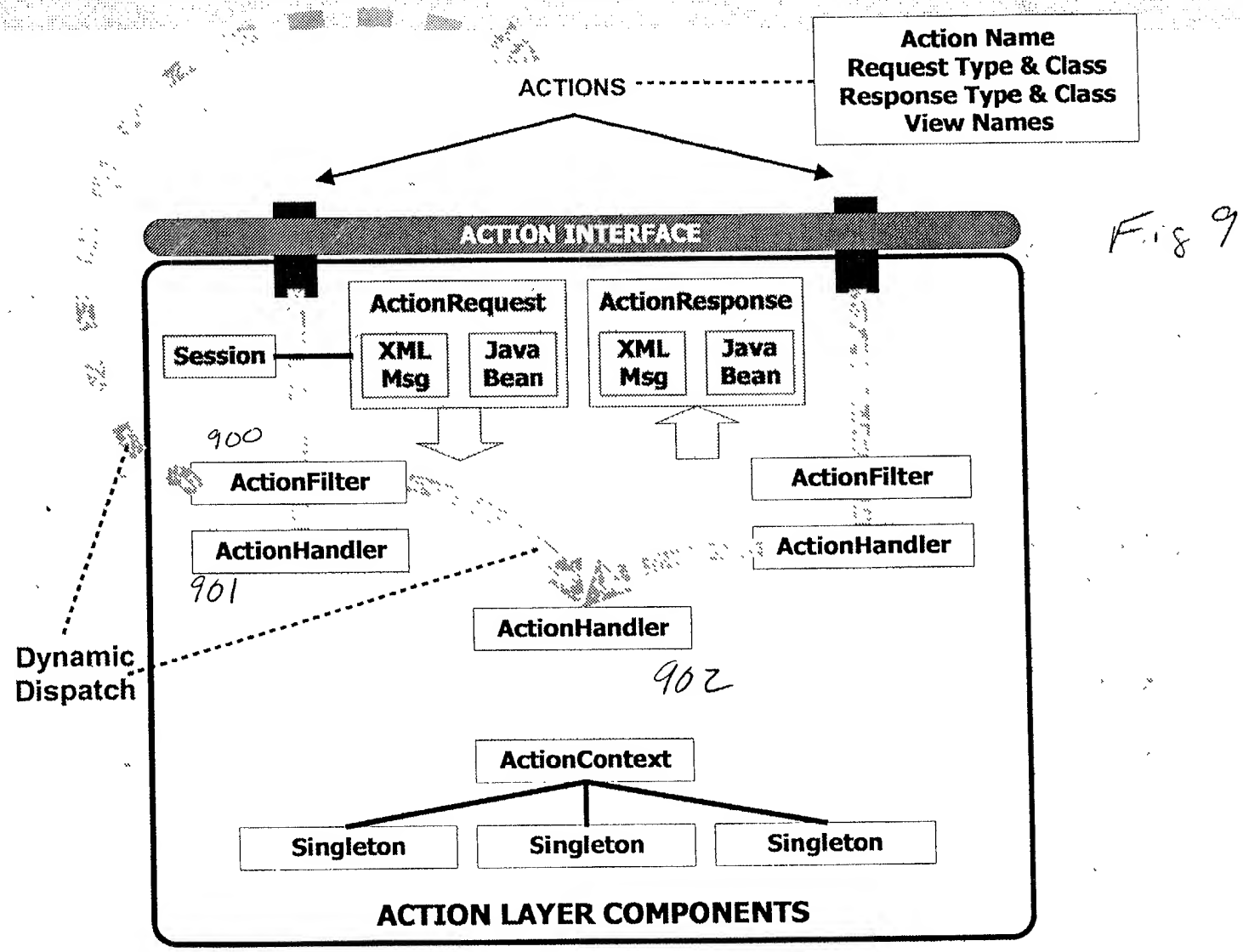


Fig 9

View Layer Components

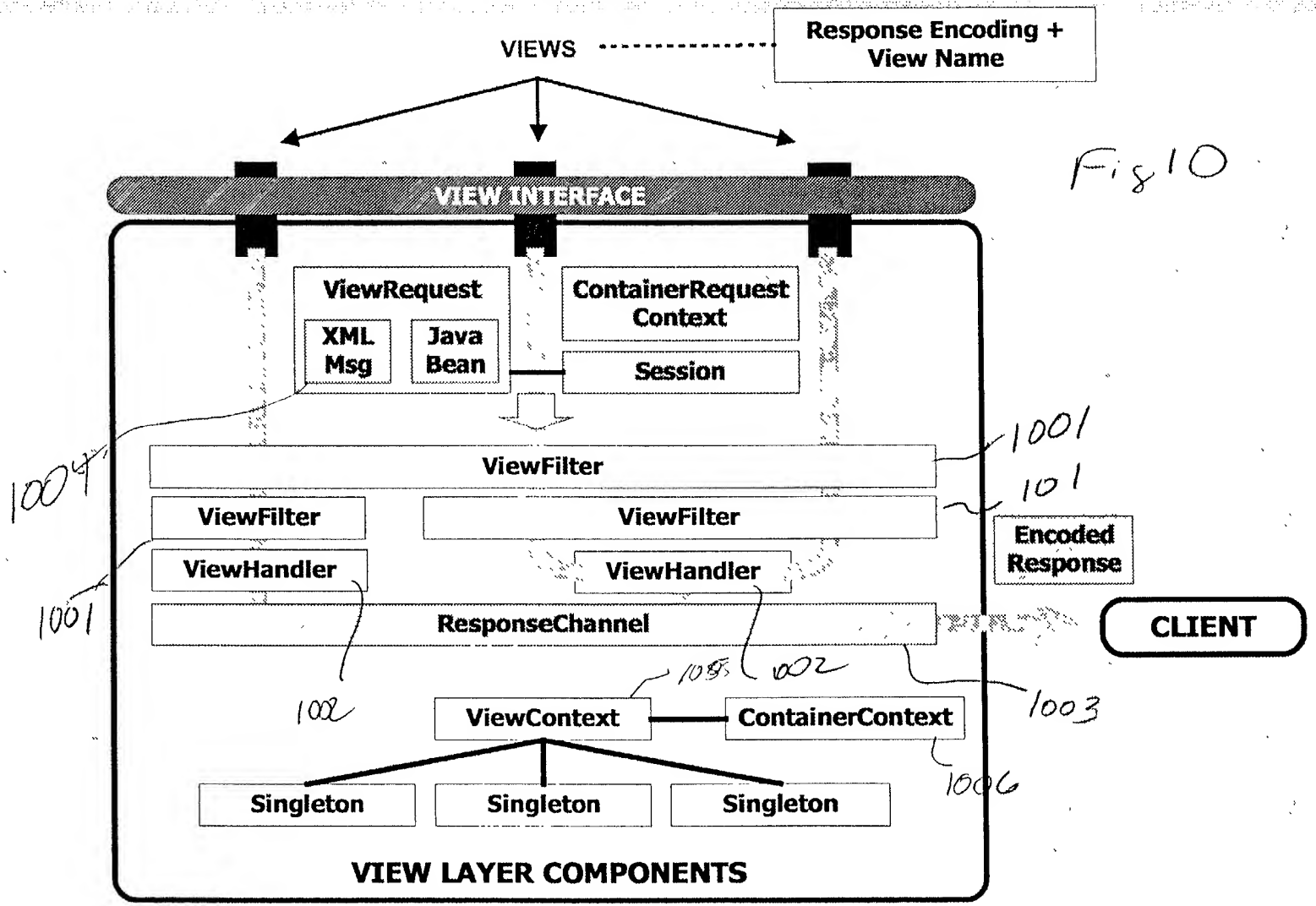
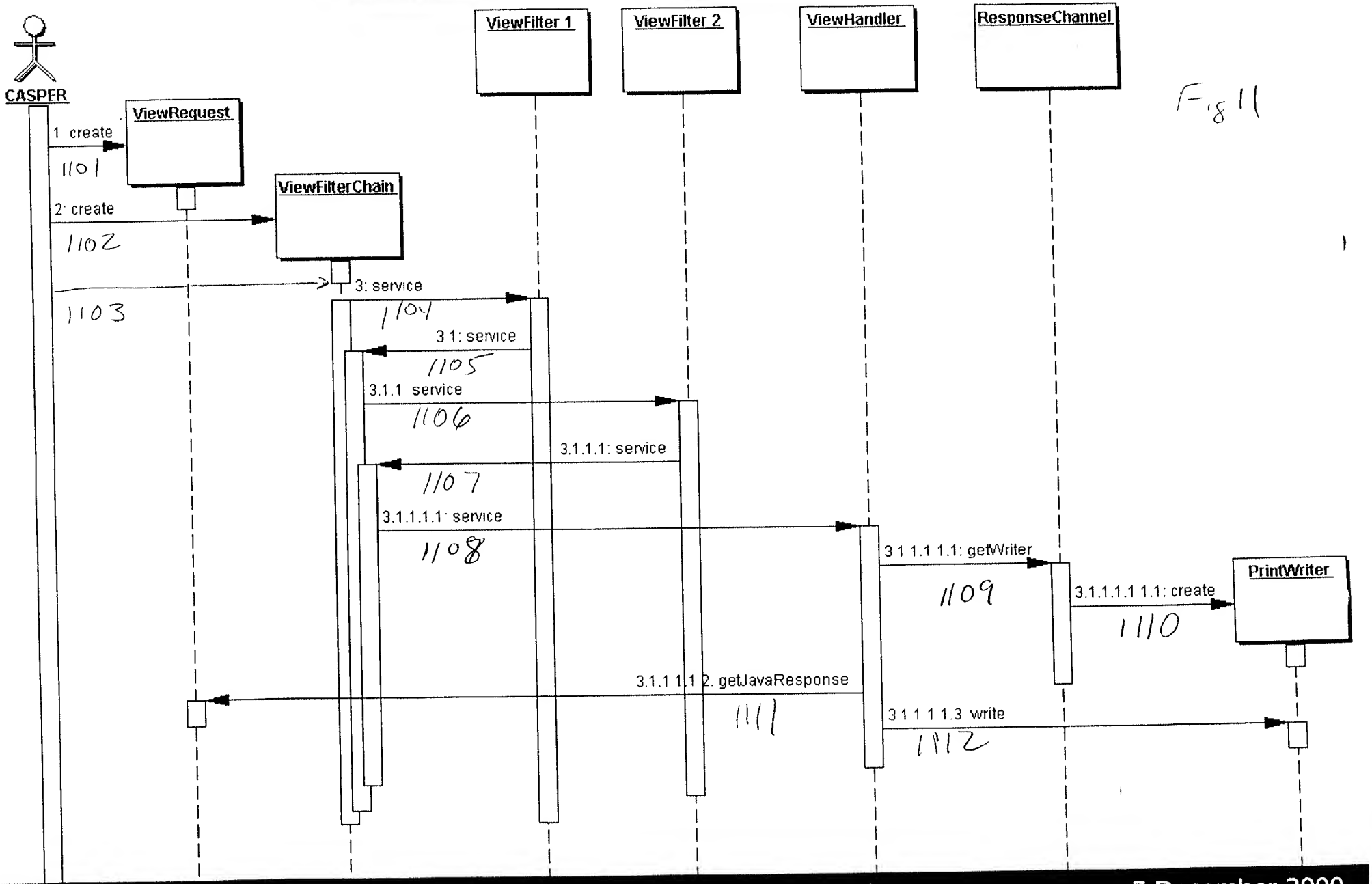
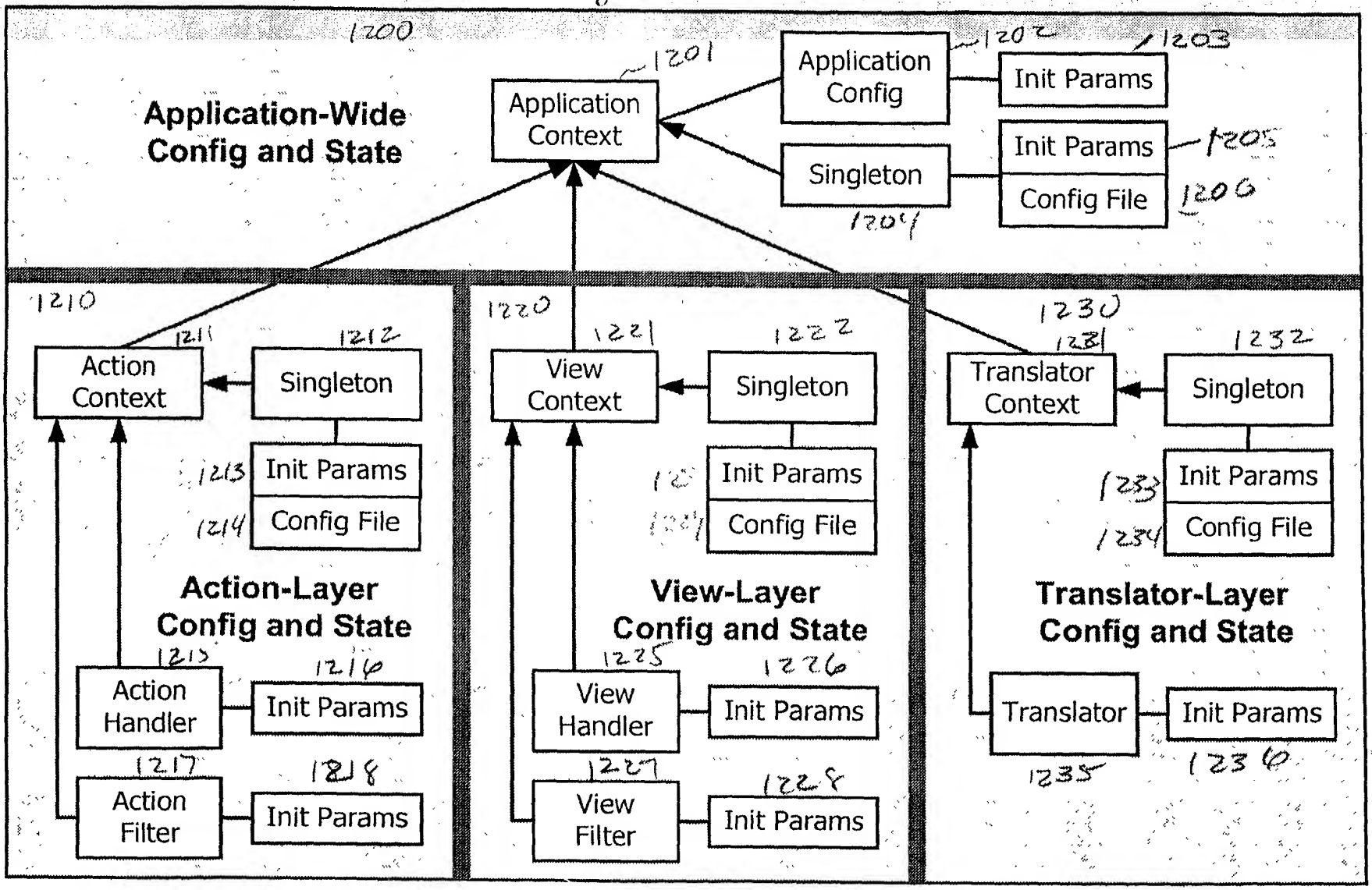


Fig 10

View Layer Request Sequence Overview

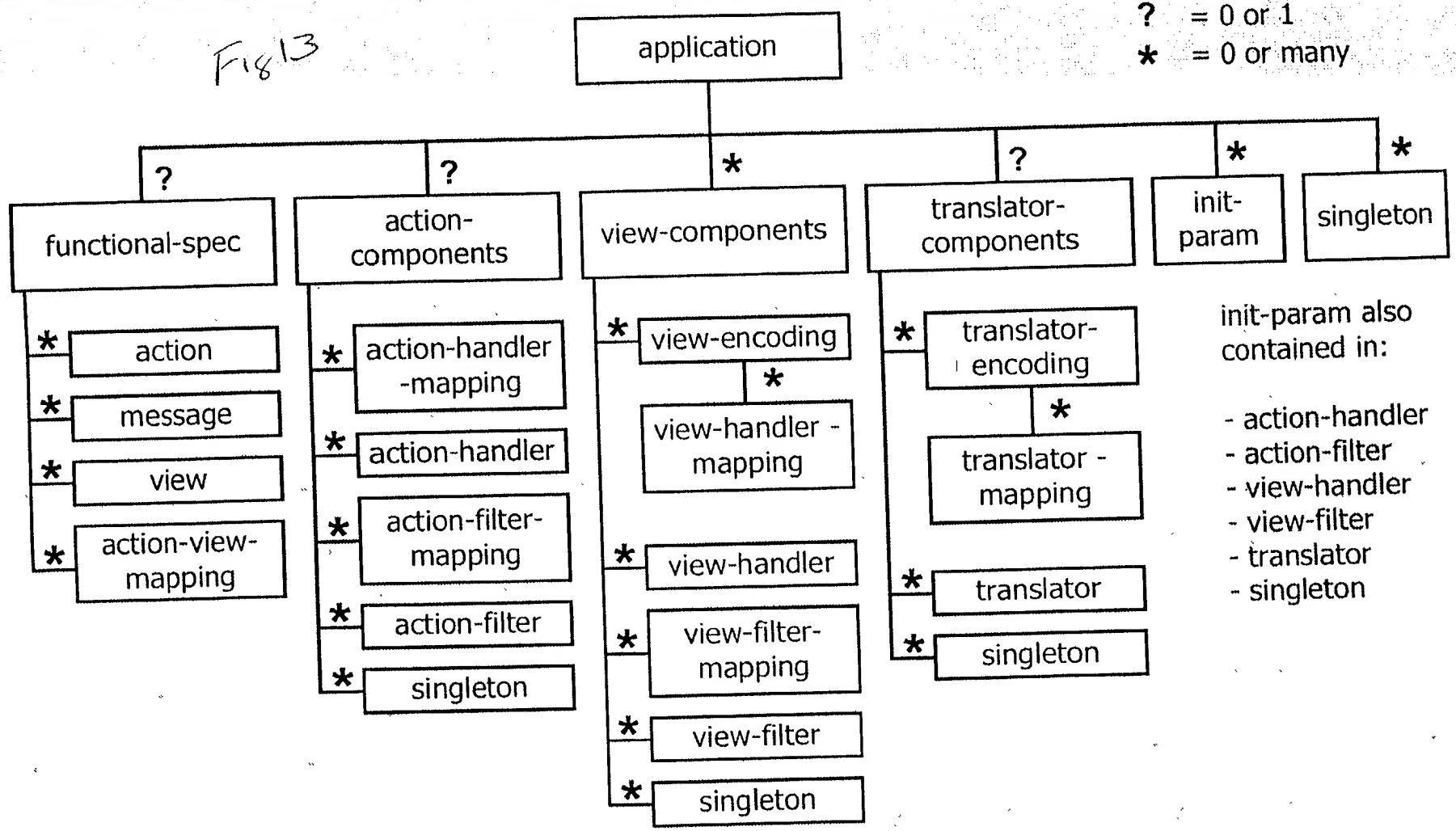


Configuration And State Architecture



casper-application.xml structure

Fig 13




```
<!DOCTYPE application
   PUBLIC "-//GE CASPER//DTD config casper-application-1.0//EN"
   "http://casper.ge.com/dtd/config/casper-application-1.0.dtd">
```

Action Table

1401

Action	application request format	application response format	Dispatcher
getproduct	get-product-req	product-description-rsp	

1402

filter
⋮
ActionHandler

Fig 14

Translator Table 1501

client request format	application request format	dispatcher
NV pair	any	

1502

Translator i

Fig 15

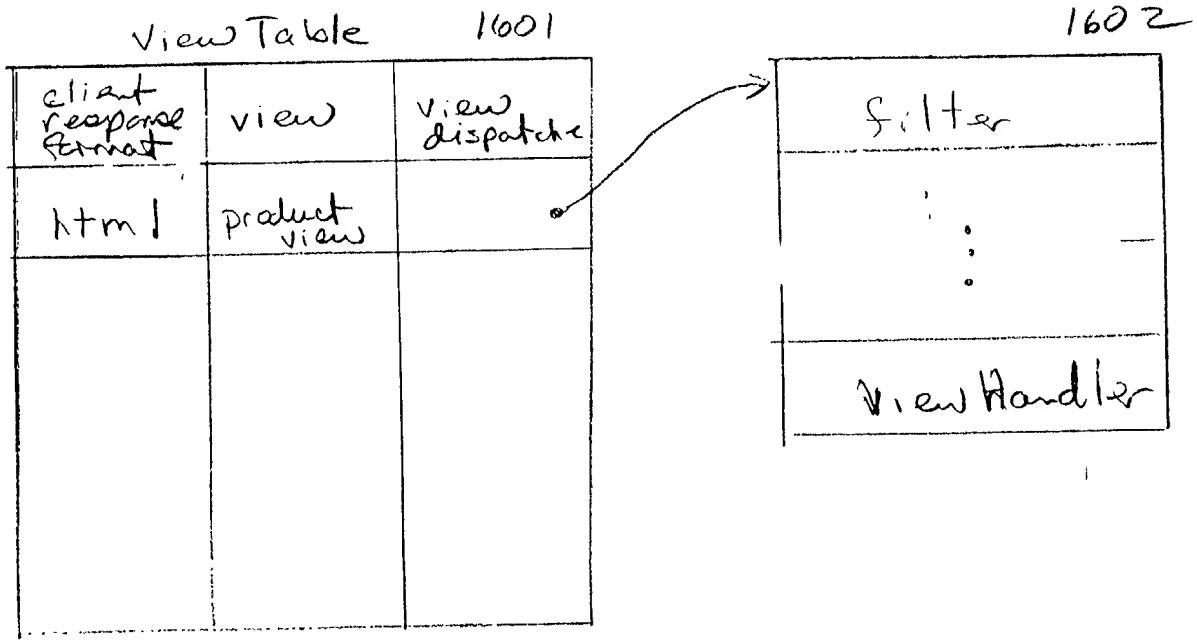

View Table
1601
client response format
view
view dispatcher
html
product view
1602
filter
View Handler

Fig 16

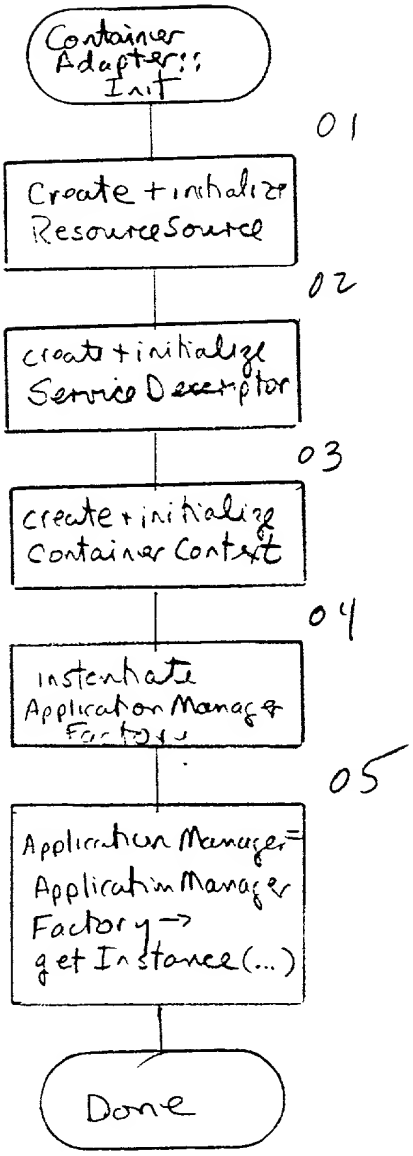
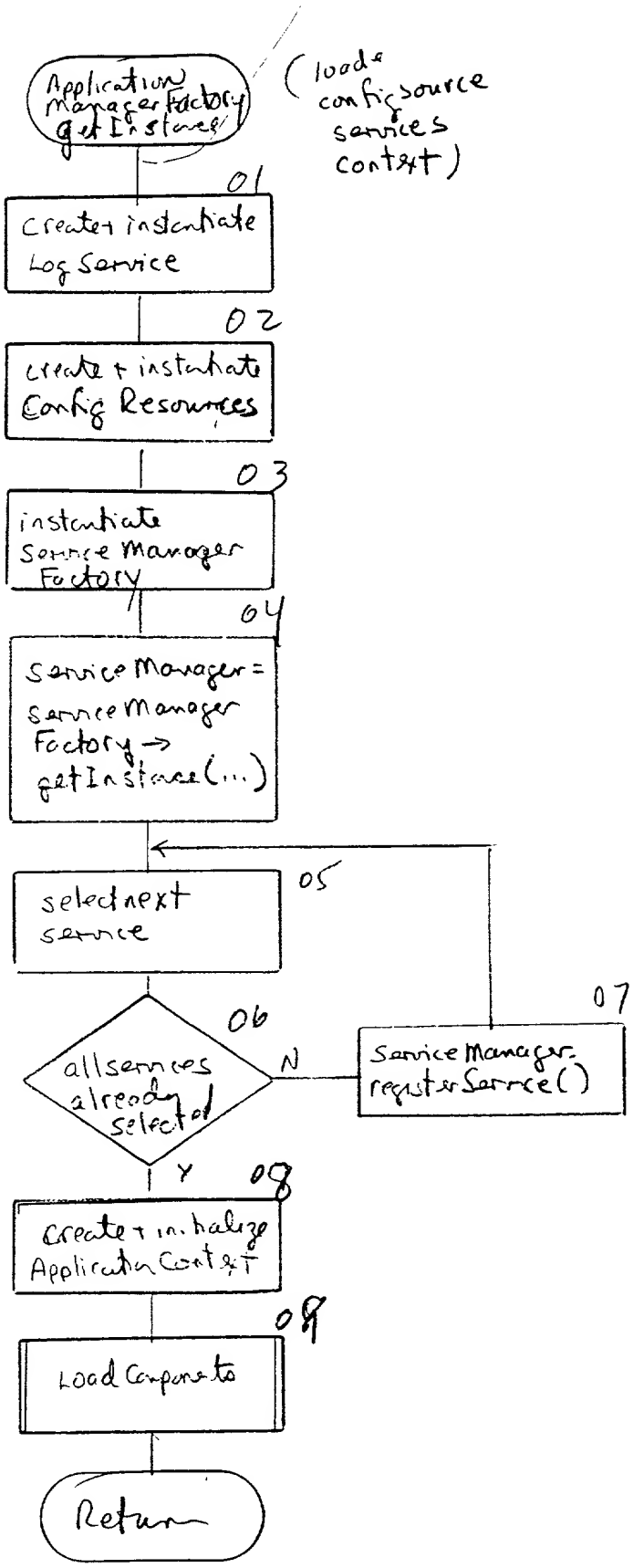

Container Adapter:: Init
01
Create + initialize ResourceSource
02
create + initialize Service Descriptor
03
create + initialize Container Context
04
instantiate Application Manager Factory
05
Application Manager = Application Manager Factory → get Instance(...)
Done

Fig 17


ApplicationManagerFactory getInstance()
(loads config source services context)
01
Create + instantiate Log Service
02
Create + instantiate Config Resources
03
instantiate Service Manager Factory
04
ServiceManager = ServiceManager Factory → getInstance(...)
05
select next service
06
all services already selected
N
07
ServiceManager. registerService()
Y
08
Create + initialize ApplicationContext
09
Load Components
Return

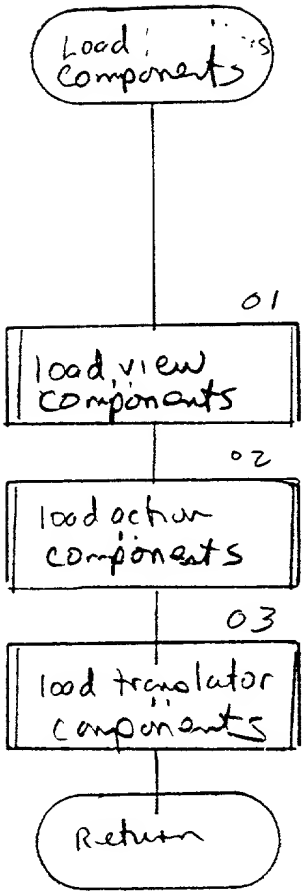

Load components
01
load view components
02
load actor components
03
load translator components
Return

FIG. 19

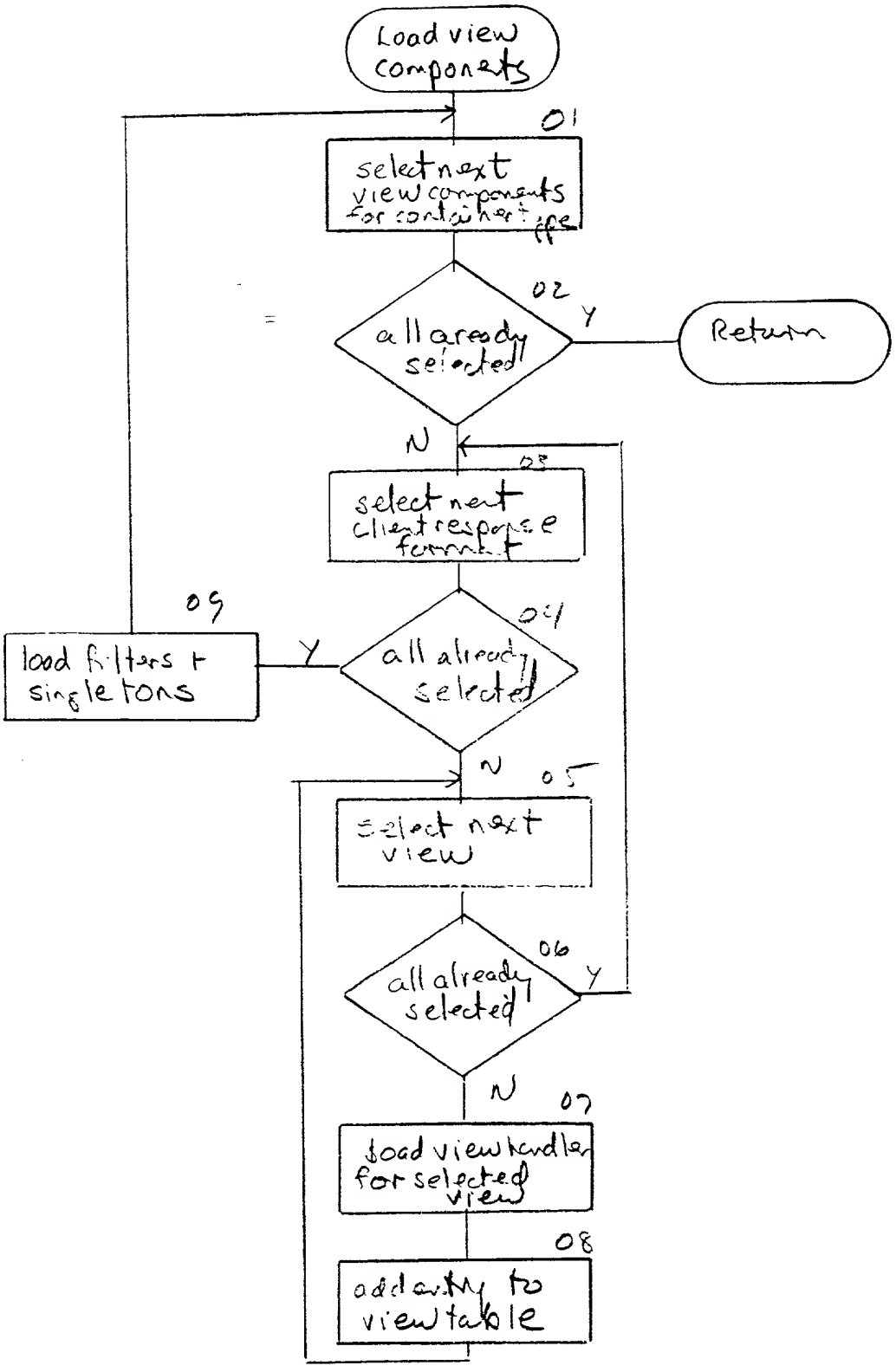

Load view components
01
select next view components for container
02
all already selected
Y
Return
N
03
select next client response format
09
load filters + singletons
Y
04
all already selected
N
05
select next view
06
all already selected
Y
N
07
load view handler for selected view
08
add entry to view table

Fig. 20

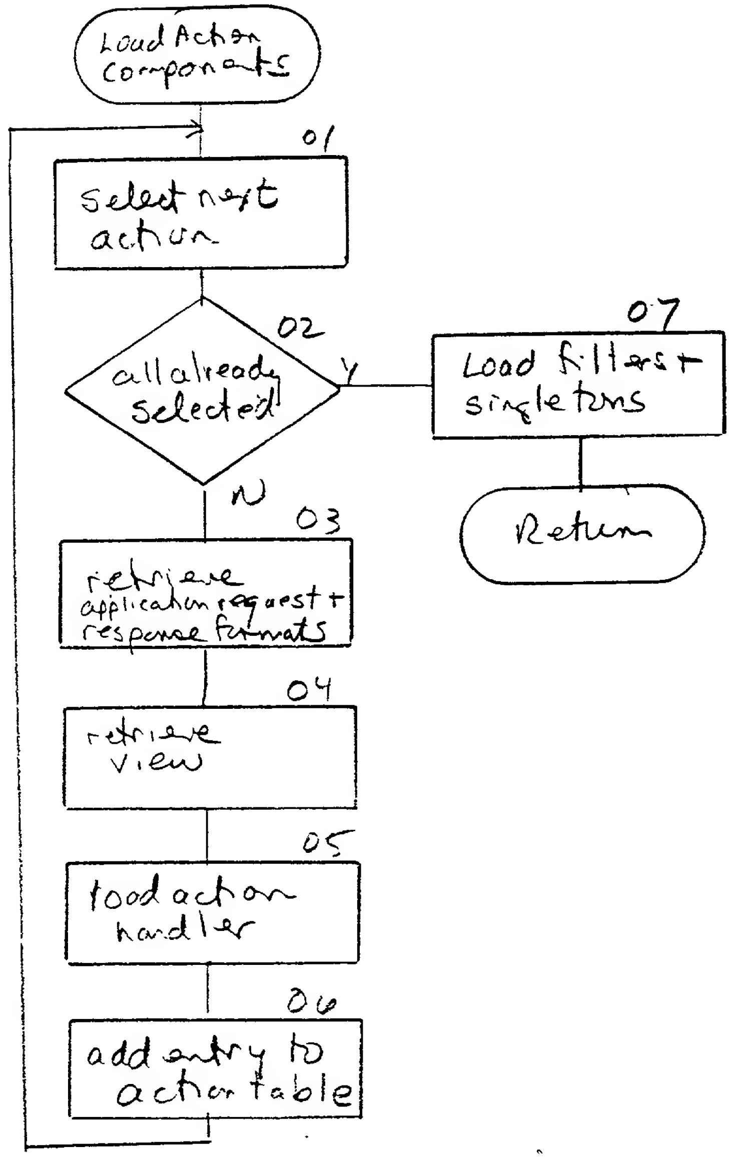


Fig. 21

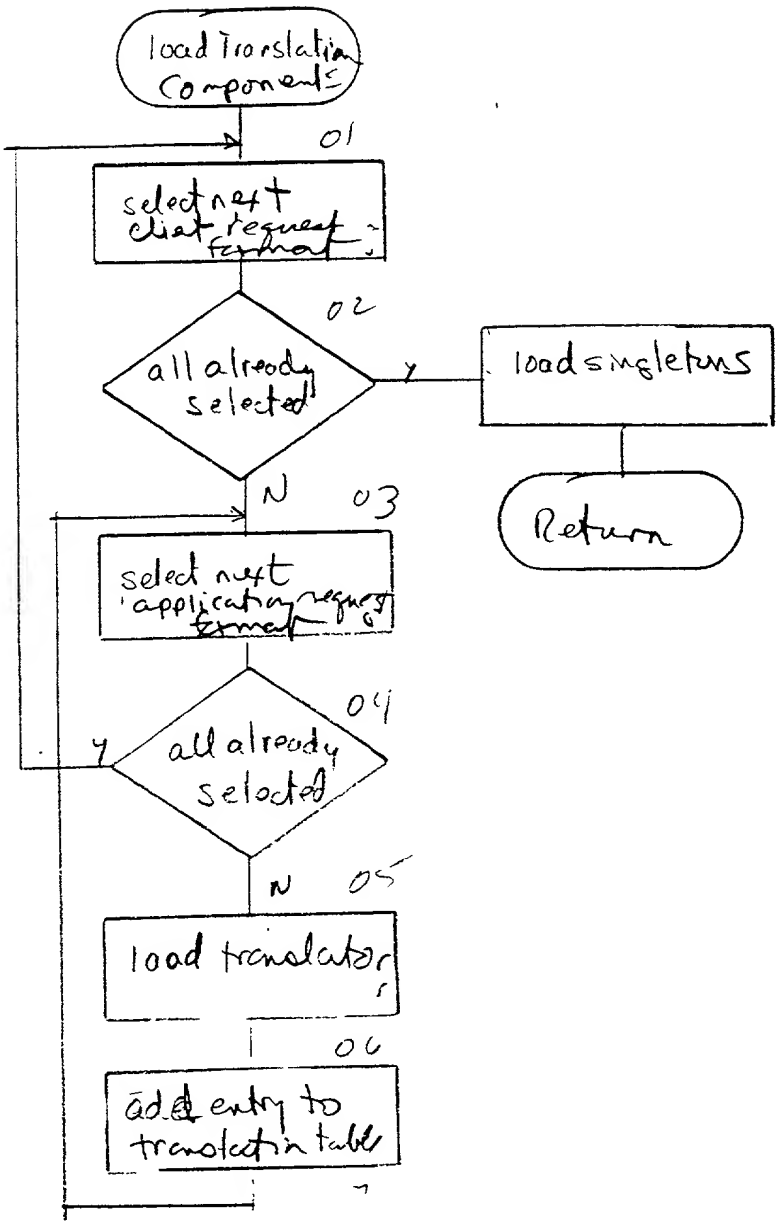

load Translation Components
01
select next client request format
02
all already selected
Y
load singletons
Return
N
03
select next application request format
04
all already selected
Y
N
05
load translator
06
add entry to translation table

Fig 22

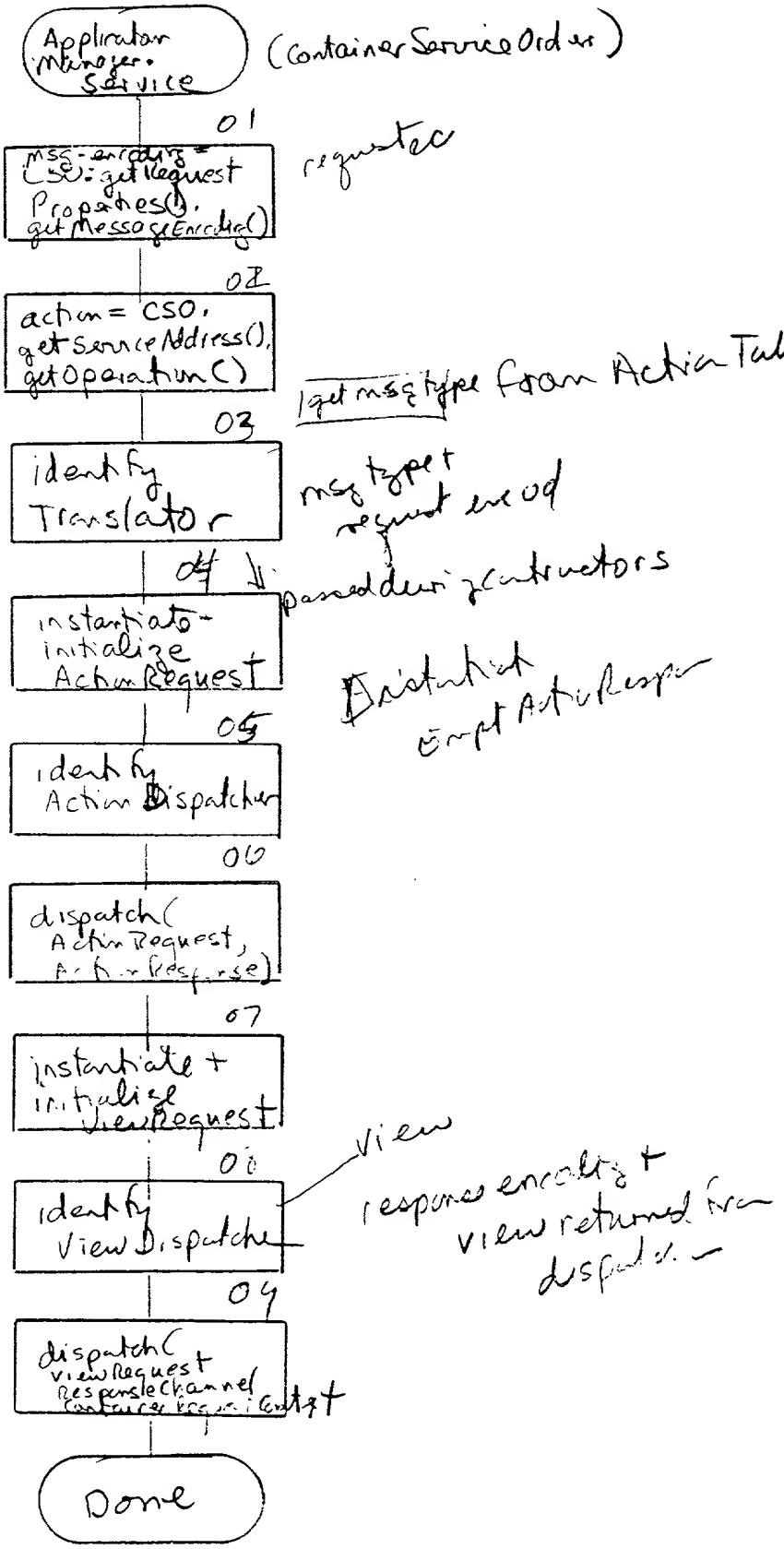

Application Manager. Service
(Container Service Order)
01
msg-encoding CSO: getRequest Properties(), getMessageEncoding()
requested
02
action = CSO. getServiceAddress(), getOperation()
get msg type from Action Table
03
Identify Translator
msg type + request encod
parsed decoding instructors
04
instantiate-initialize ActionRequest
Distinct except ActionResp
05
Identify Action Dispatcher
06
dispatch(ActionRequest, ActionResponse)
07
instantiate + initialize ViewRequest
08
Identify View Dispatcher
view
response encoding + view returned from dispatch
09
dispatch(ViewRequest ResponseChannel Container request context
Done

Fig. 23

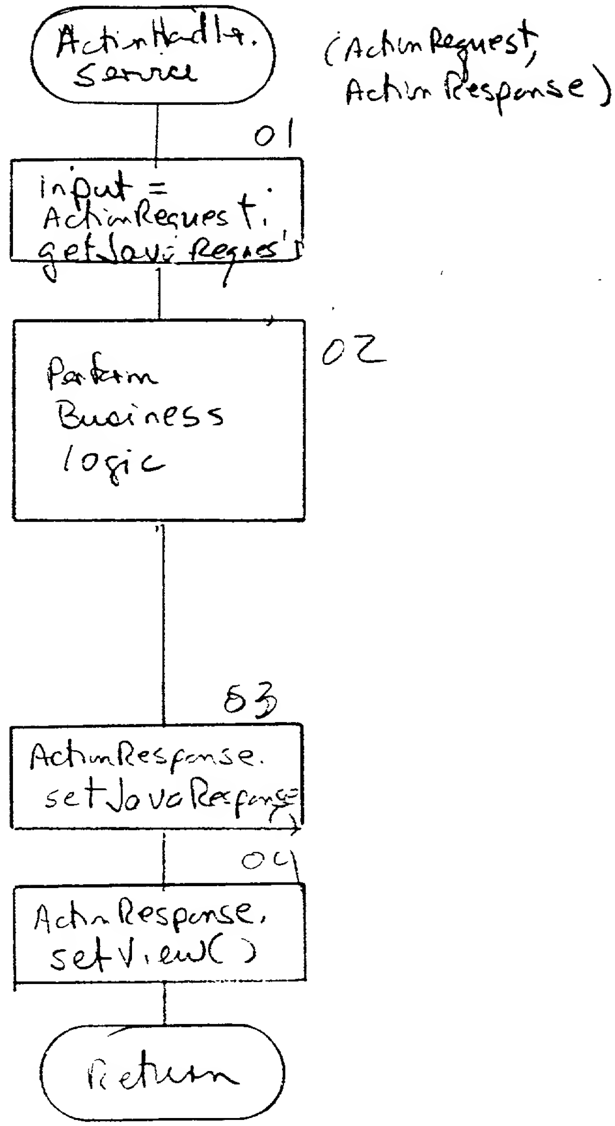


Fig. 24

Service Framework Overview

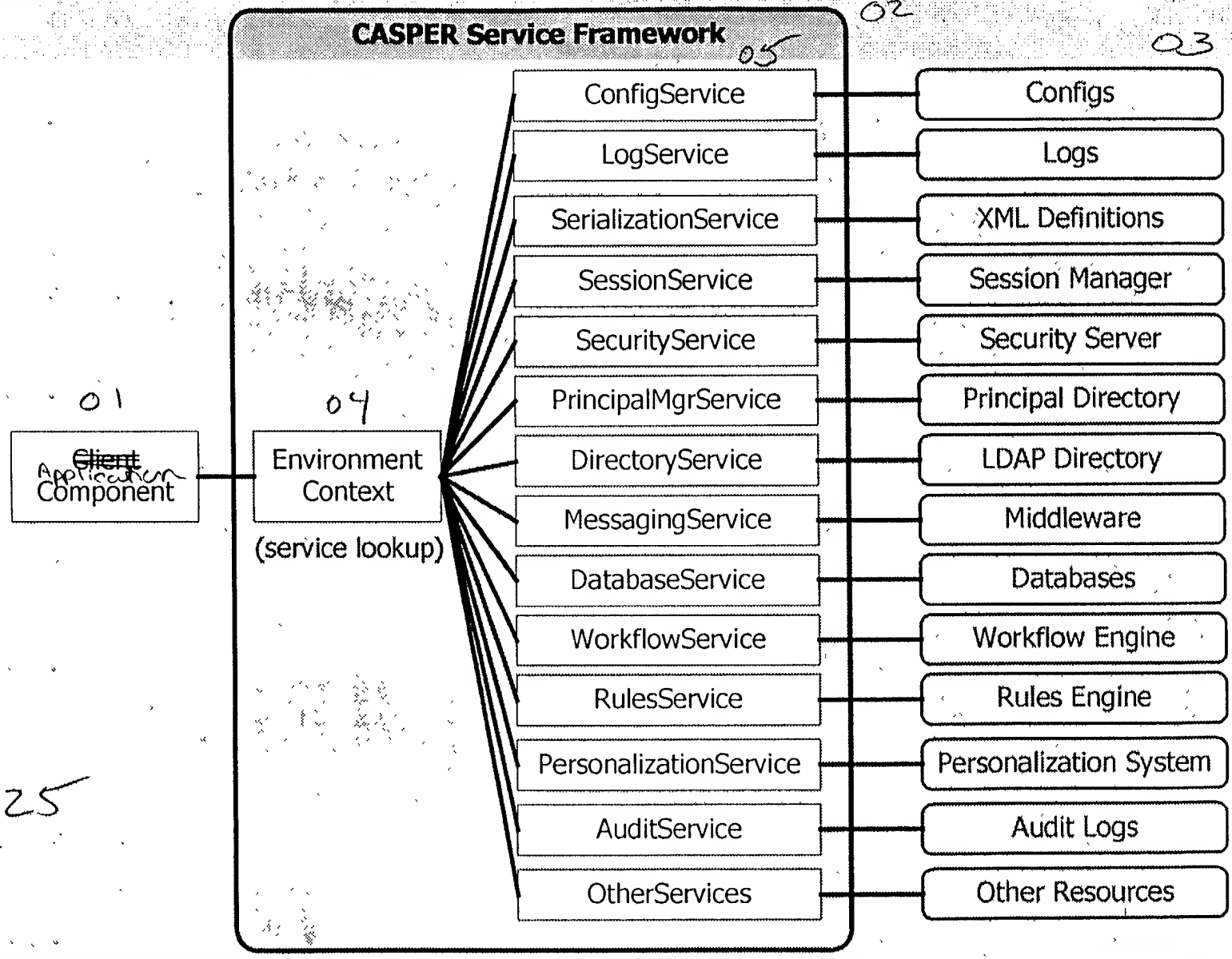


Fig 25

Service Classes & Interfaces

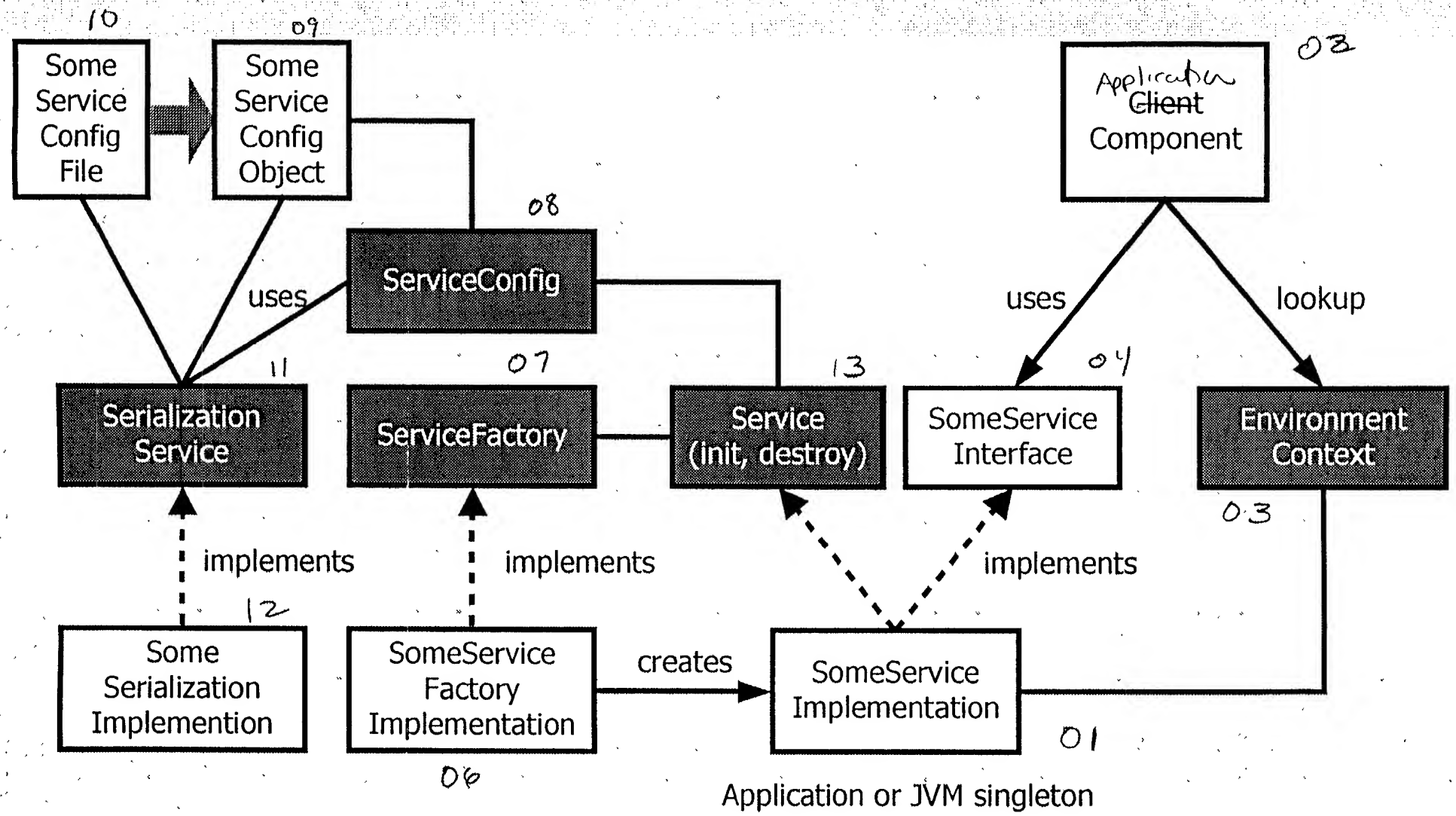


Fig 26

Service Framework Configuration

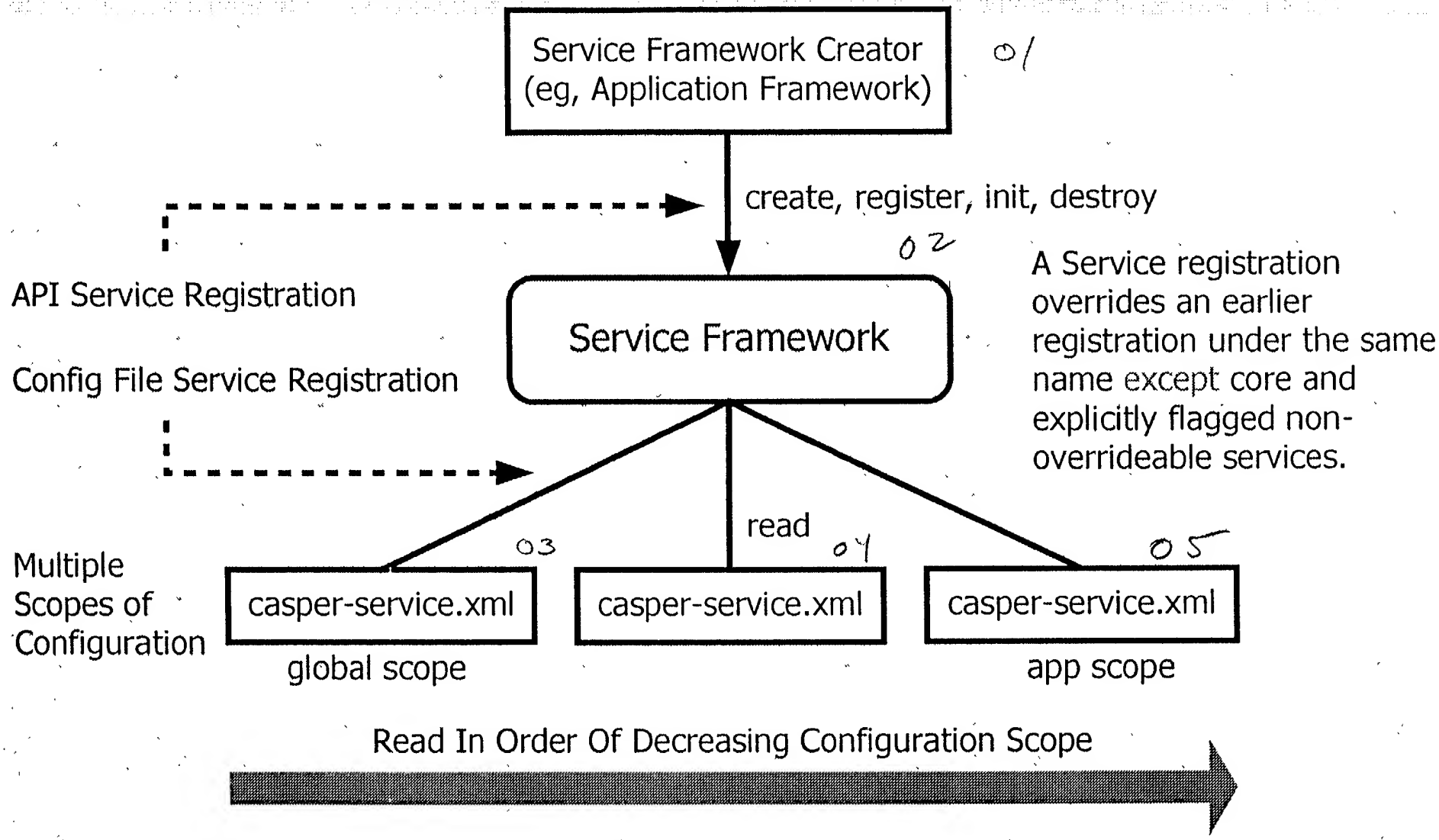


Fig 27

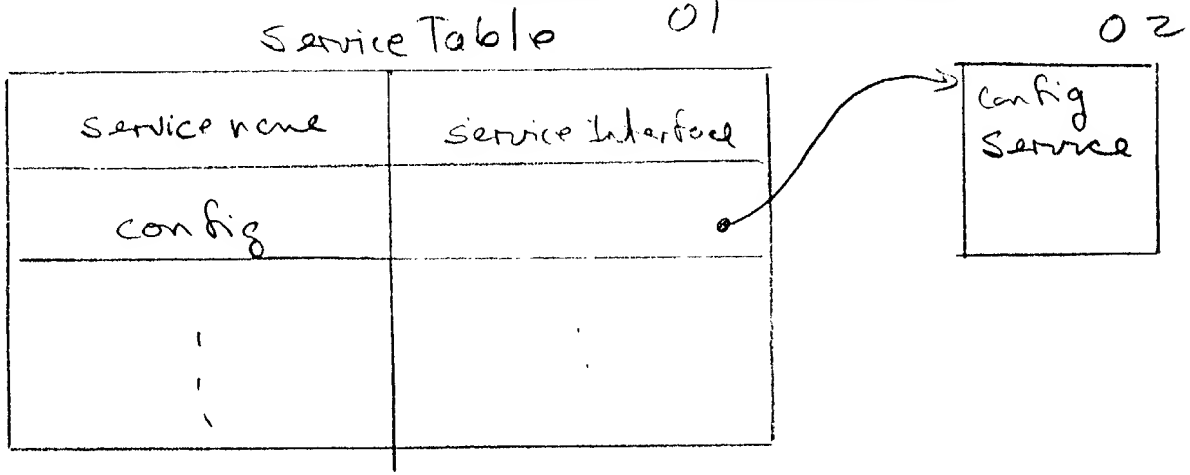


Fig 28

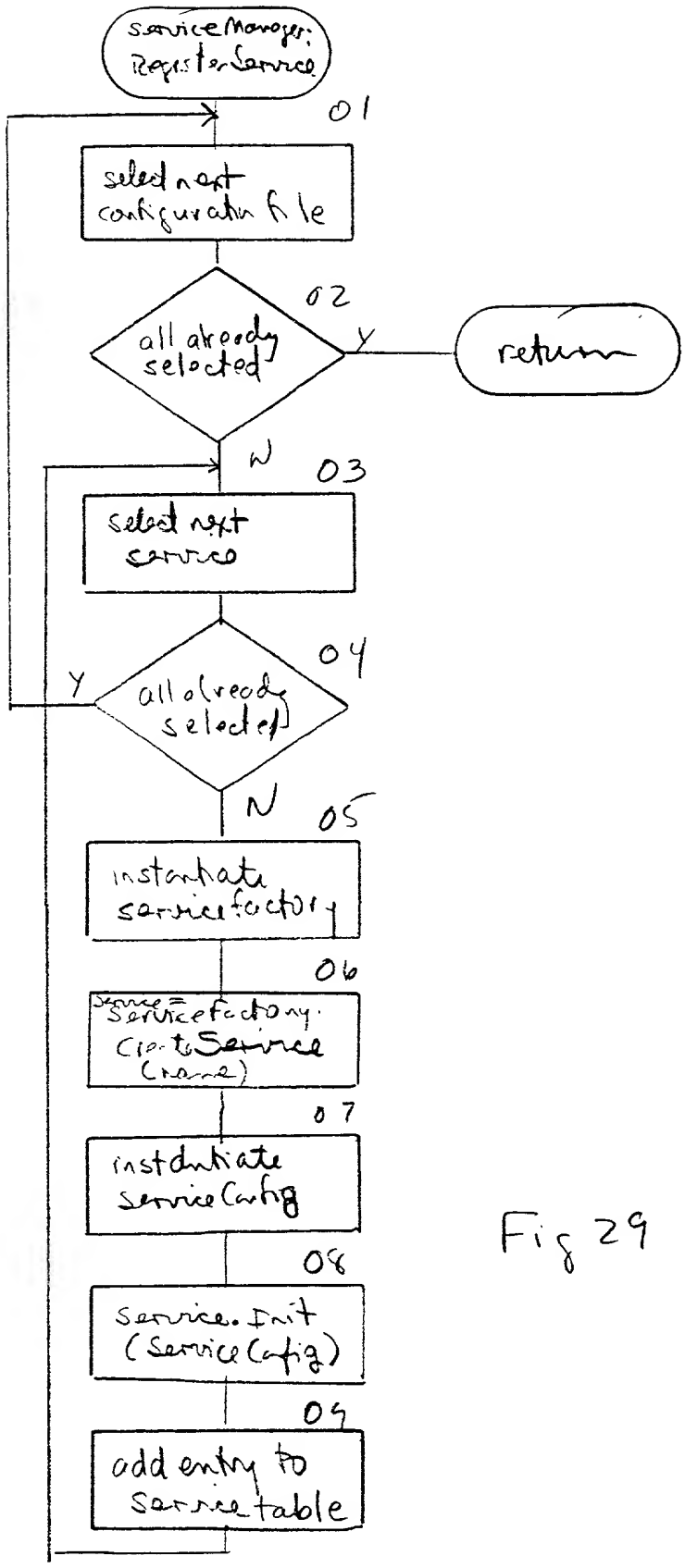

ServiceManager:
RegisterService
01
select next
configuration file
02
all already
selected
Y
return
N
03
select next
service
04
all already
selected
Y
N
05
instantiate
service factory
06
ServiceFactory:
CreateService
(name)
07
instantiate
ServiceConfig
08
Service.Init
(ServiceConfig)
09
add entry to
service table

Fig 29

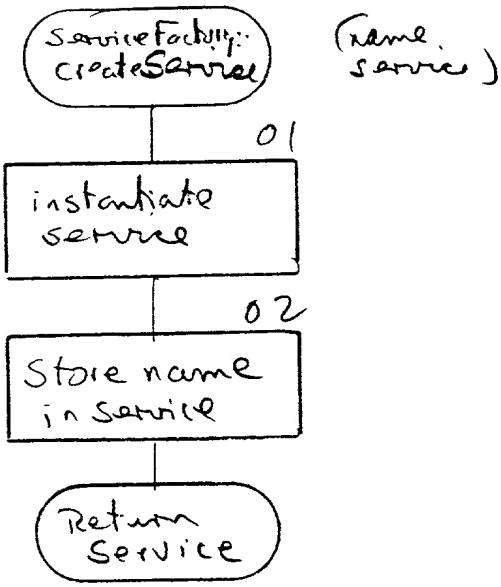


Fig 30

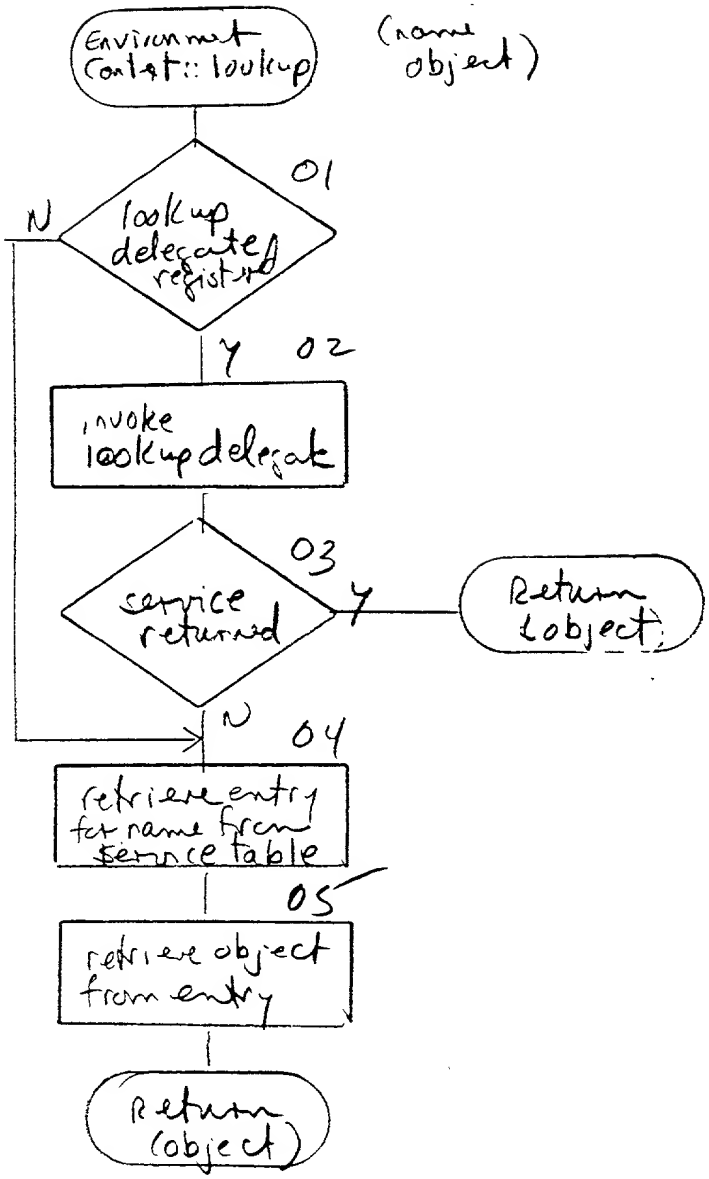
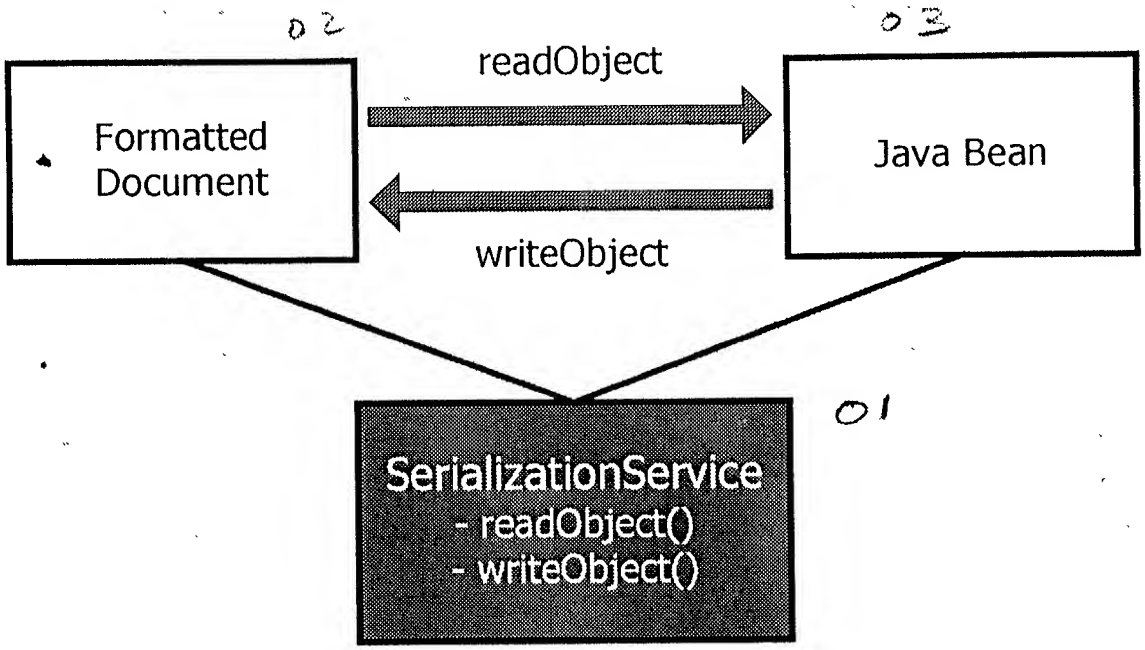

Environment Context::lookup
(name object)
01
lookup delegate registered
N
Y
02
invoke lookup delegate
03
service returned
Y
Return (object)
N
04
retrieve entry for name from service table
05
retrieve object from entry
Return (object)

Fig 31

Serialization Service



A SerializationService is responsible for transforming and validating formatted documents to and from JavaBean objects

FIG. 32

Configuration Definition with SFO XML Tools

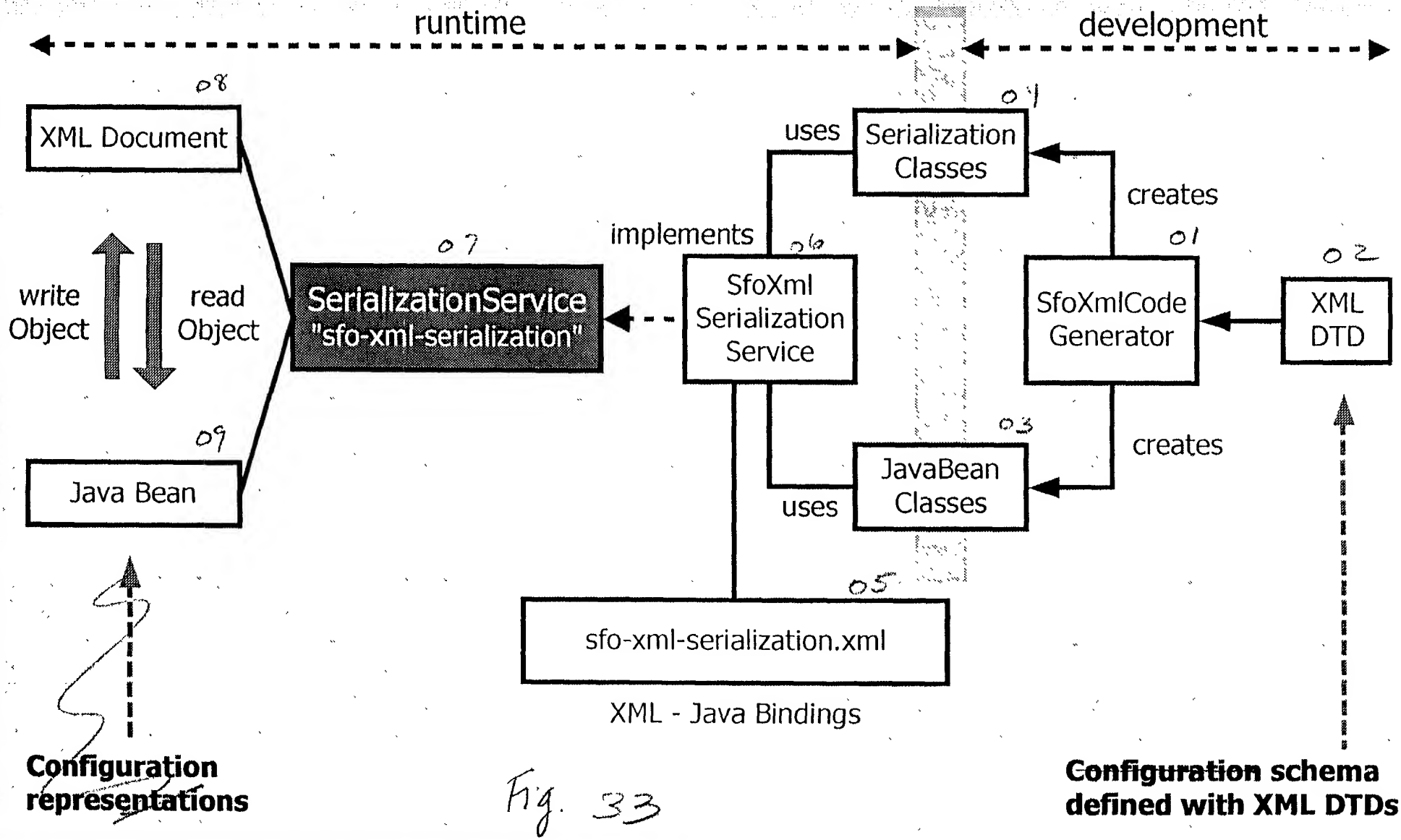


Fig. 33

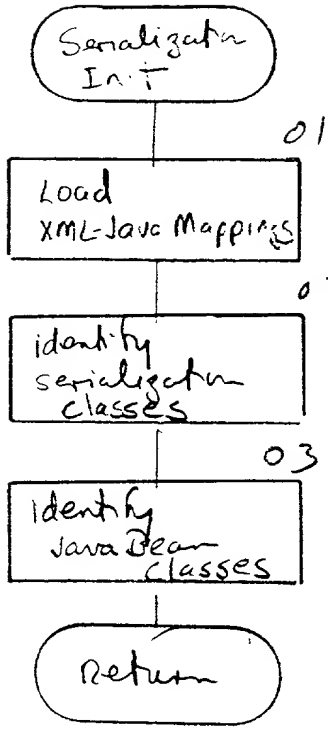

Serialization Init
01
Load XML-Java Mappings
02
identify serialization classes
03
Identify Java Bean classes
Return

Fig. 34

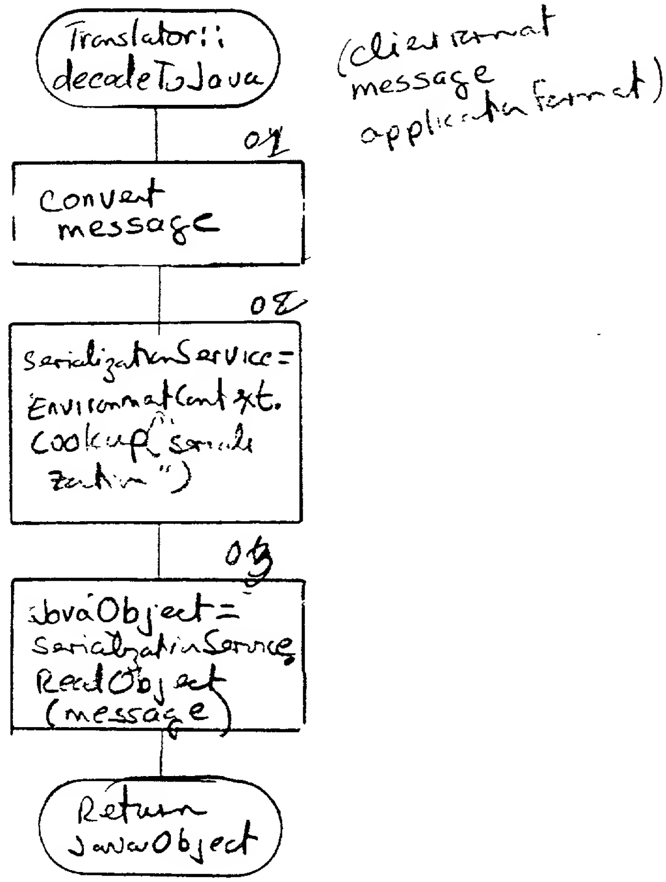


Fig. 35

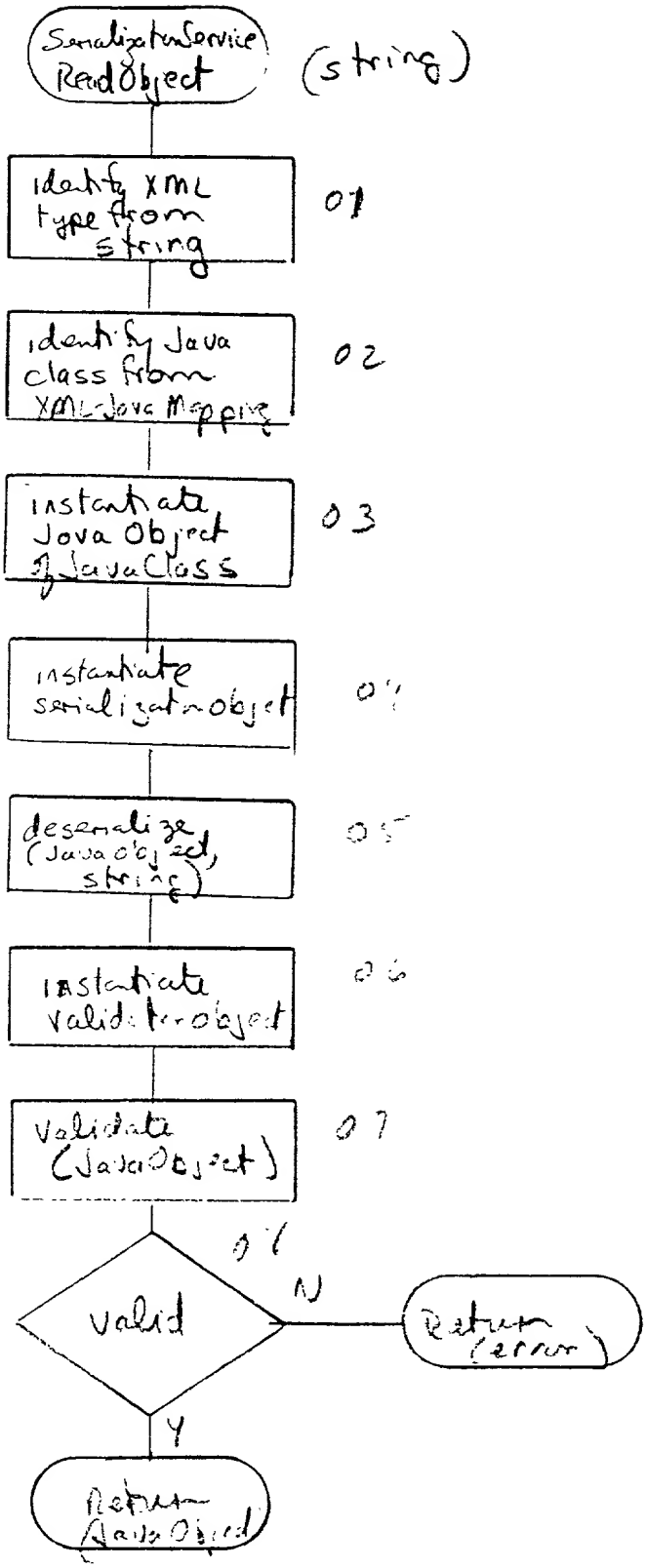

SerializationService
ReadObject
(string)
identify XML type from string
01
identify Java class from XML-Java Mapping
02
instantiate Java Object of JavaClass
03
instantiate serialization object
04
deserialize (Java object, string)
05
instantiate validation object
06
Validate (JavaObject)
07
valid
08
N
Return (error)
Y
Return (Java Object)

Fig. 36

Configuration Service

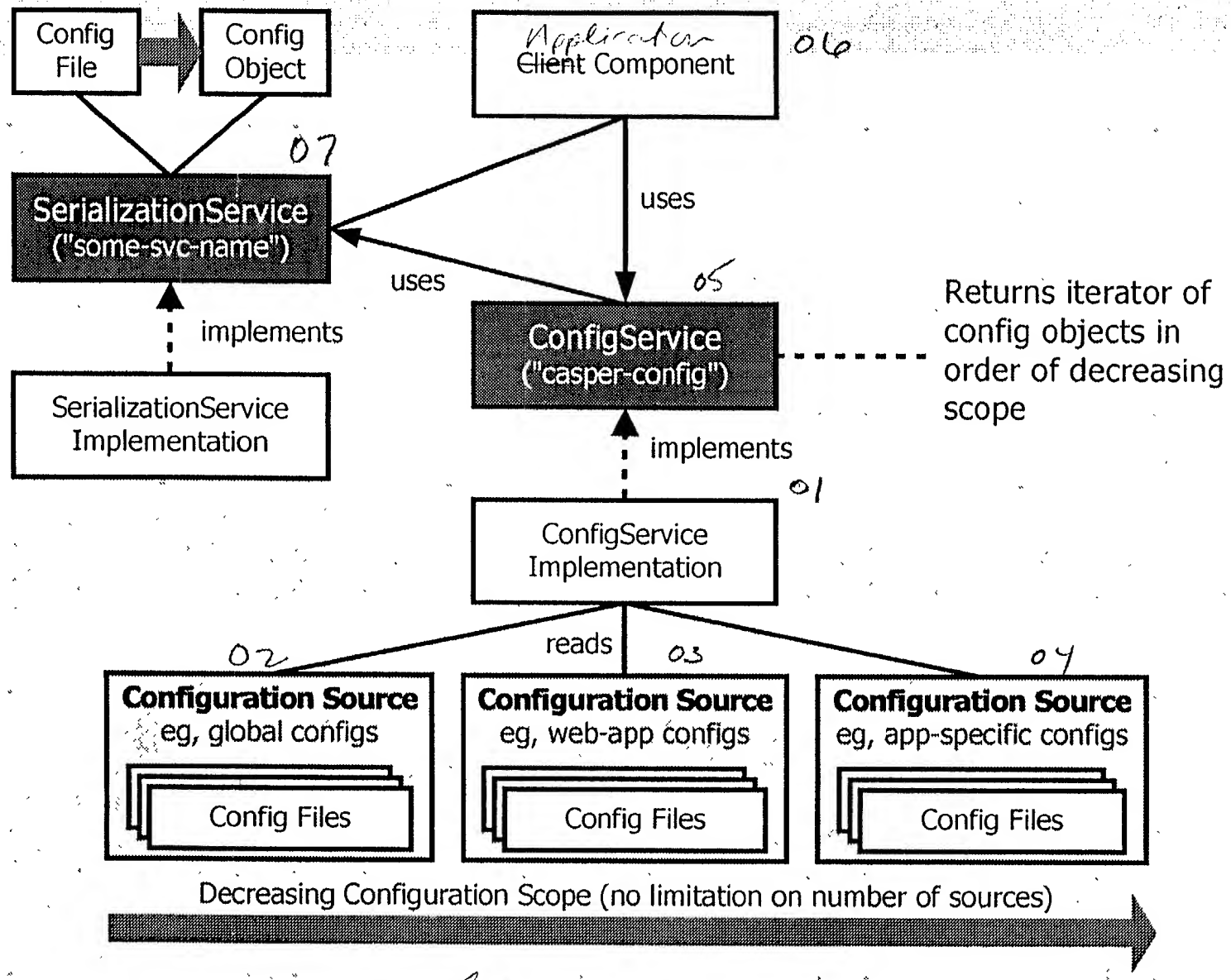


Fig. 37

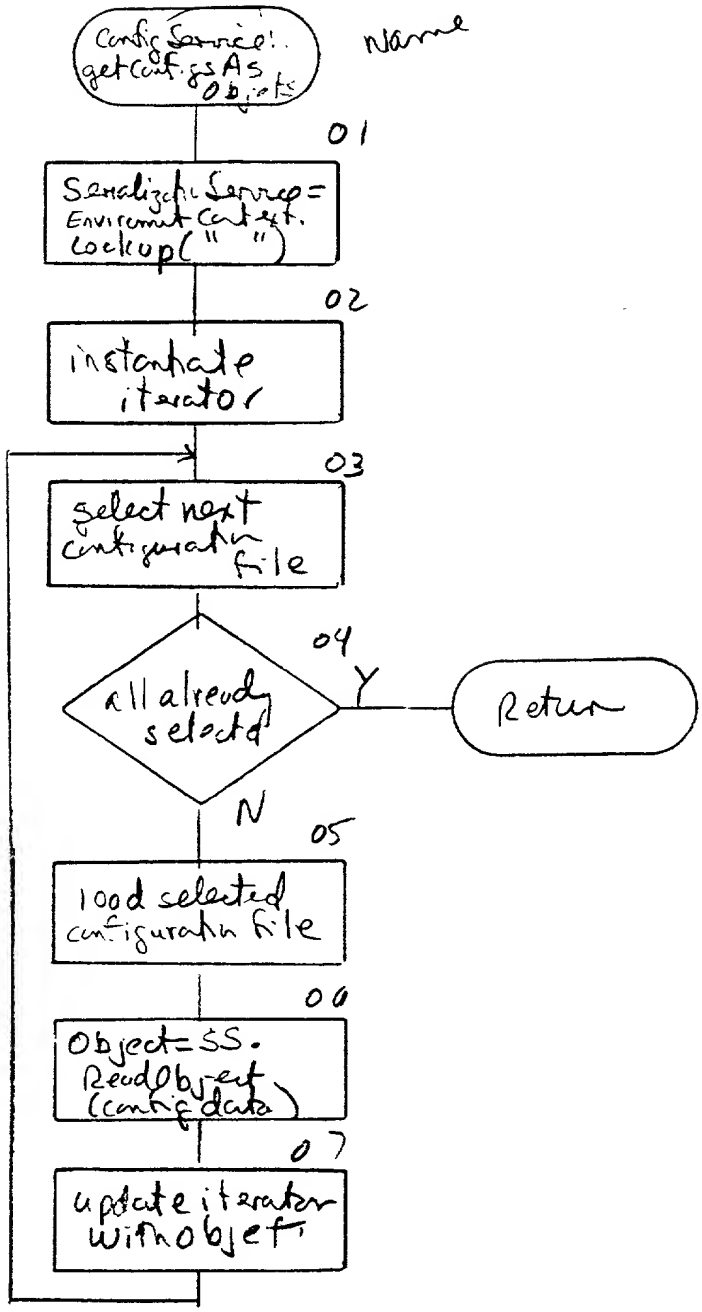

ConfigService::
getConfigsAs
Objects
Name
01
SerializableServer=
EnvironmentContext.
Lookup(" ")
02
instantiate
iterator
03
select next
configuration file
04
all already
selected
Y
Return
N
05
load selected
configuration file
06
Object=SS.
ReadObject
(config data)
07
update iterator
with object

Fig. 38

Fig. 39

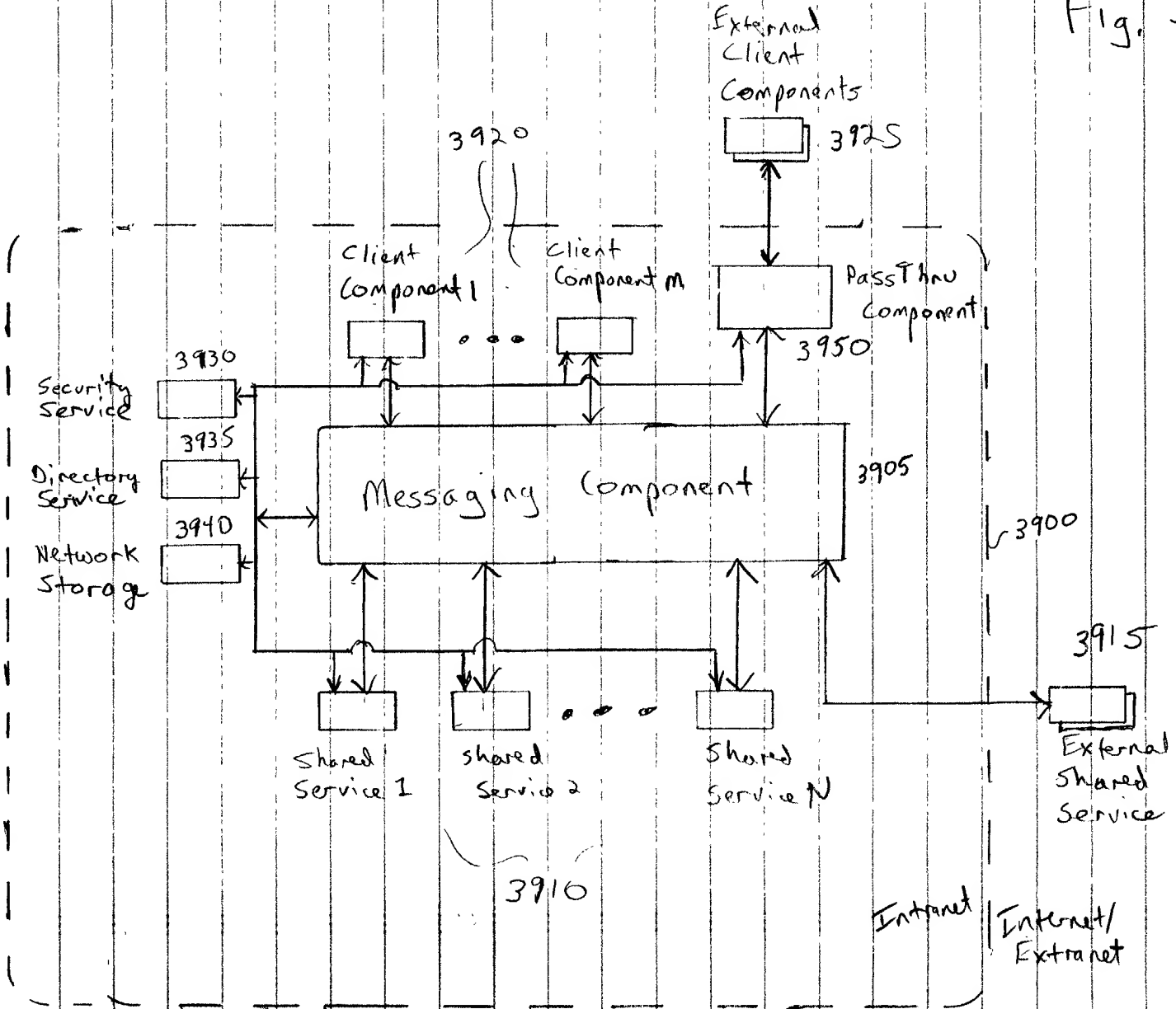


Fig. 40

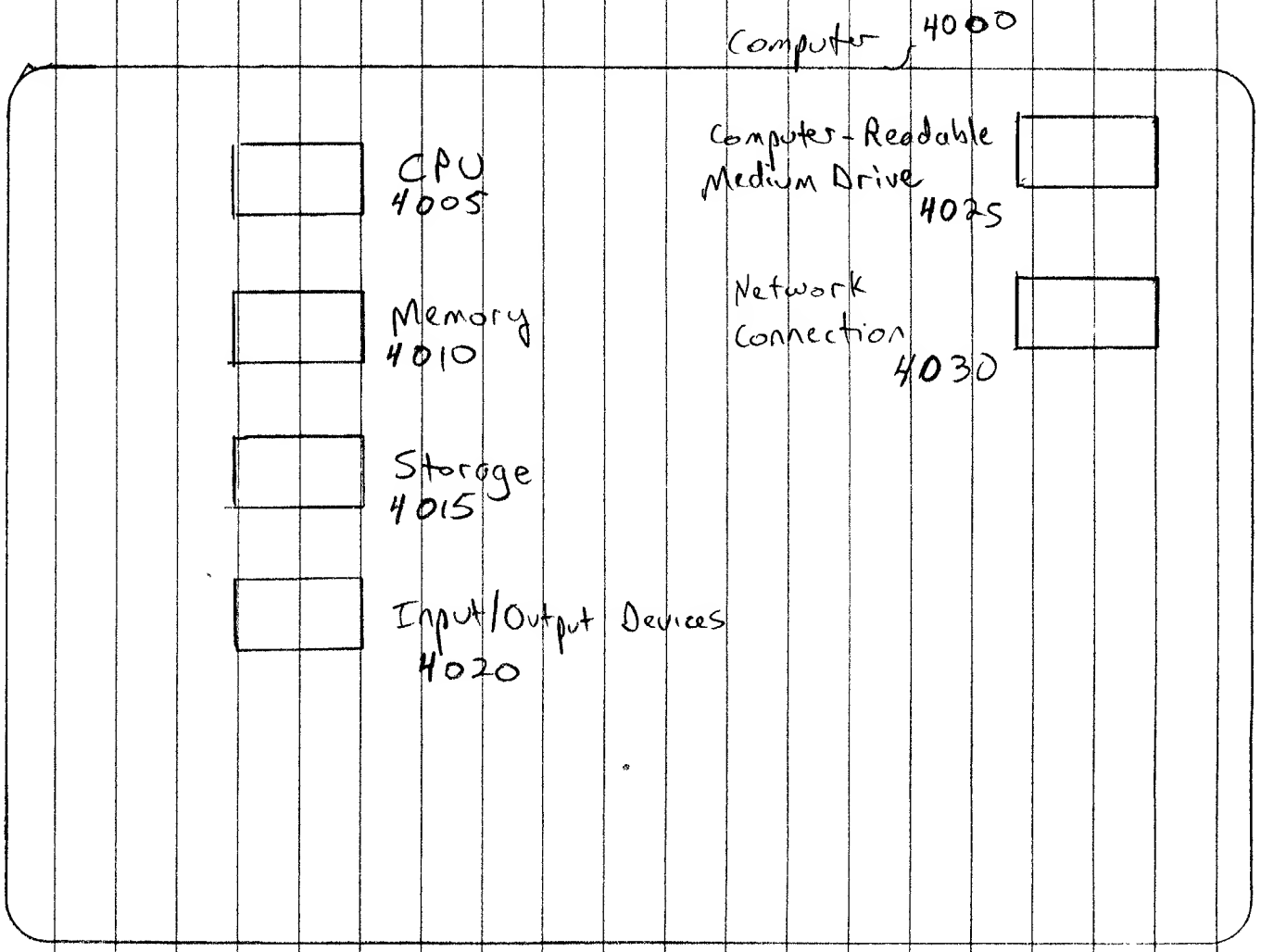
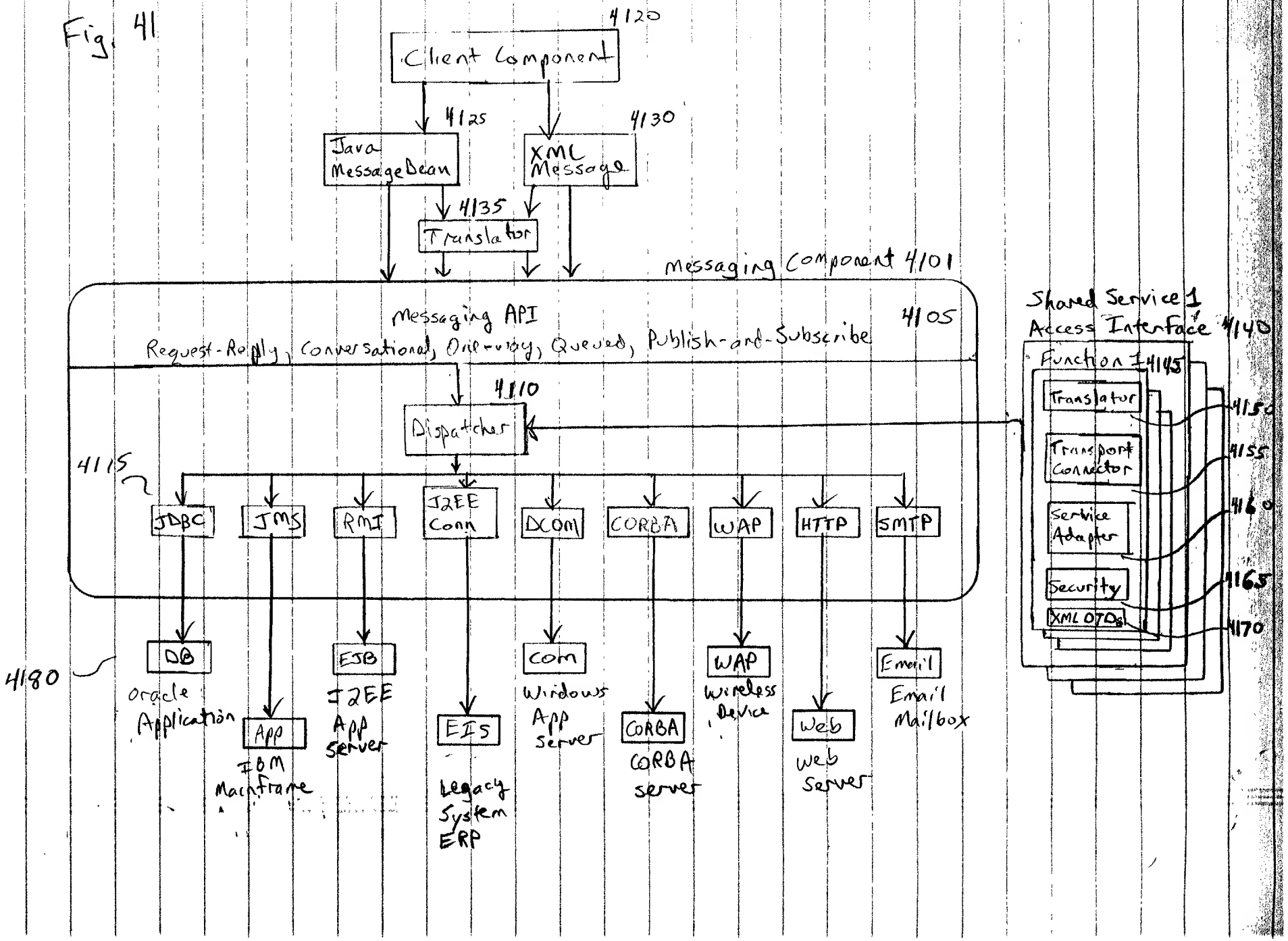

Computer 4000
CPU 4005
Memory 4010
Storage 4015
Input/Output Devices 4020
Computer-Readable Medium Drive 4025
Network Connection 4030

Fig. 41


4120
Client Component
4125
Java MessageBean
4130
XML Message
4135
Translator
Messaging Component 4101
Messaging API
4105
Request-Reply, Conversational, One-way, Queued, Publish-and-Subscribe
4110
Dispatcher
4115
JDBC
JMS
RMI
J2EE Conn
DCOM
CORBA
WAP
HTTP
SMTP
Shared Service 1
Access Interface
4140
Function 1
4145
Translator
4150
Transport Connector
4155
Service Adapter
4160
Security
4165
XML OTO
4170
4180
DB
Oracle Application
App
IBM Mainframe
EJB
J2EE App Server
EIS
Legacy System ERP
com
Windows App Server
CORBA
CORBA server
WAP
Wireless Device
Web
Web Server
Email
Email Mailbox

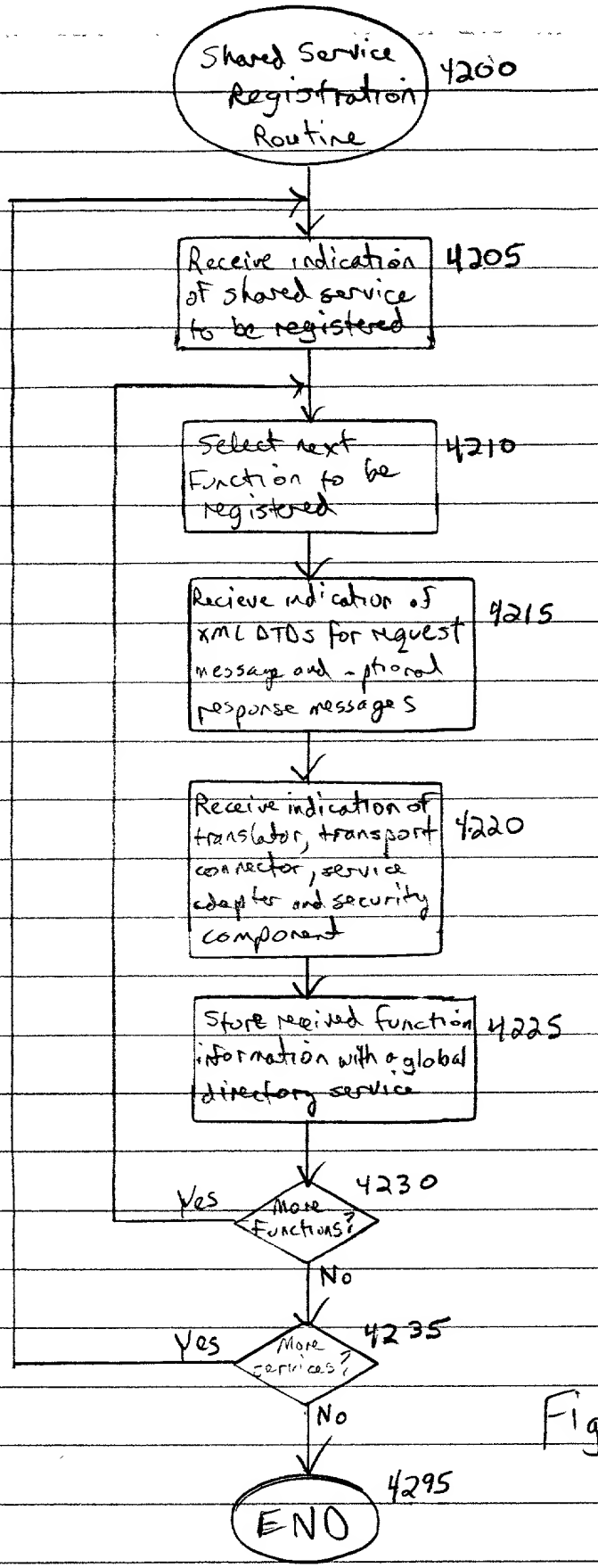

Shared Service Registration Routine
4200
Receive indication of shared service to be registered
4205
Select next Function to be registered
4210
Recieve indication of xML DTDs for request message and optional response messages
4215
Receive indication of translator, transport connector, service adapter and security component
4220
Store received function information with a global directory service
4225
More Functions?
4230
Yes
No
More services?
4235
Yes
No
END
4295

Fig. 42

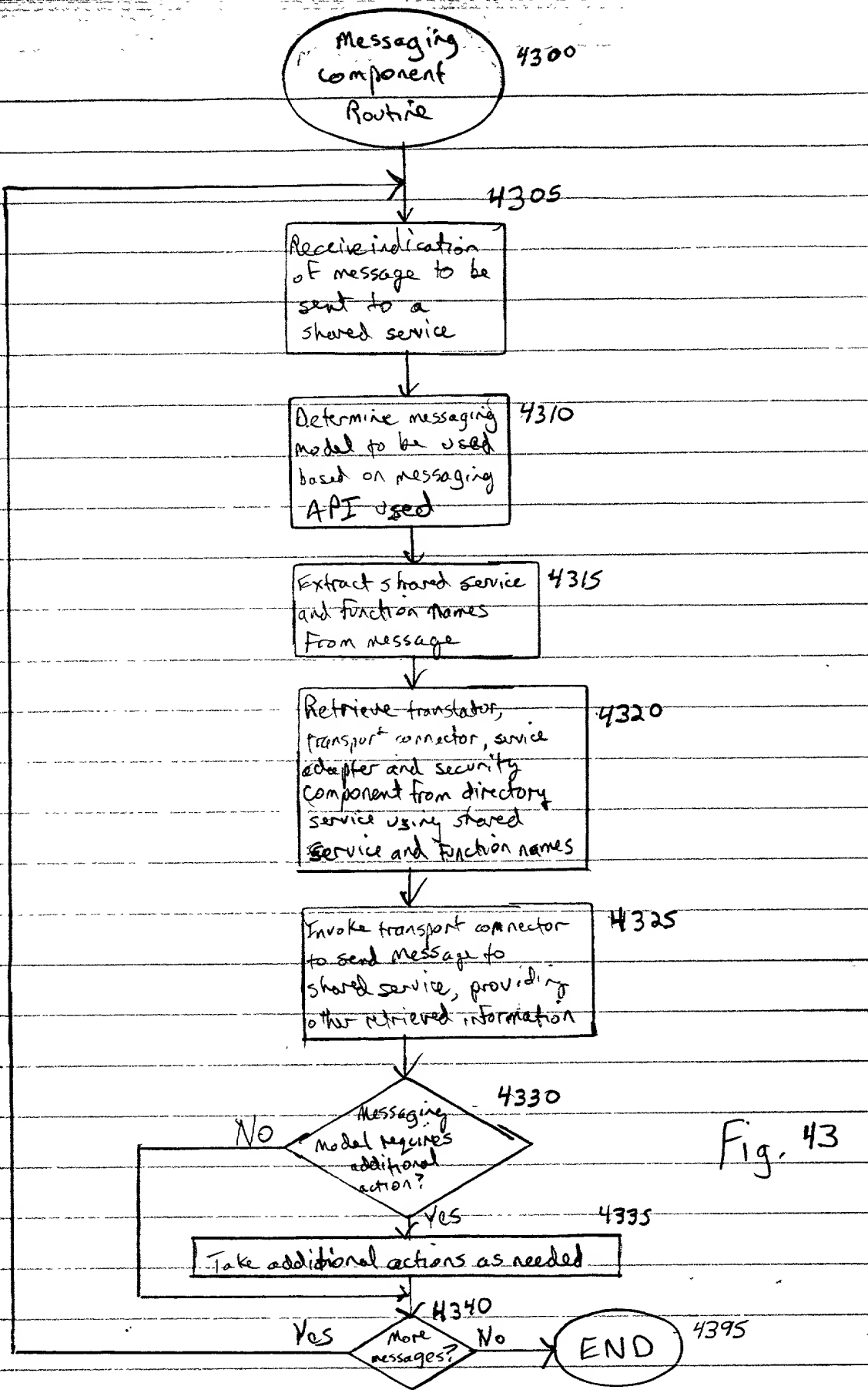

Messaging component Routine
4300
4305
Receive indication of message to be sent to a shared service
Determine messaging model to be used based on messaging API used
4310
Extract shared service and function names from message
4315
Retrieve translator, transport connector, service adapter and security component from directory service using shared service and function names
4320
Invoke transport connector to send message to shared service, providing other retrieved information
4325
Messaging model requires additional action?
4330
No
Yes
Take additional actions as needed
4335
More messages?
4340
Yes
No
END
4395

Fig. 43

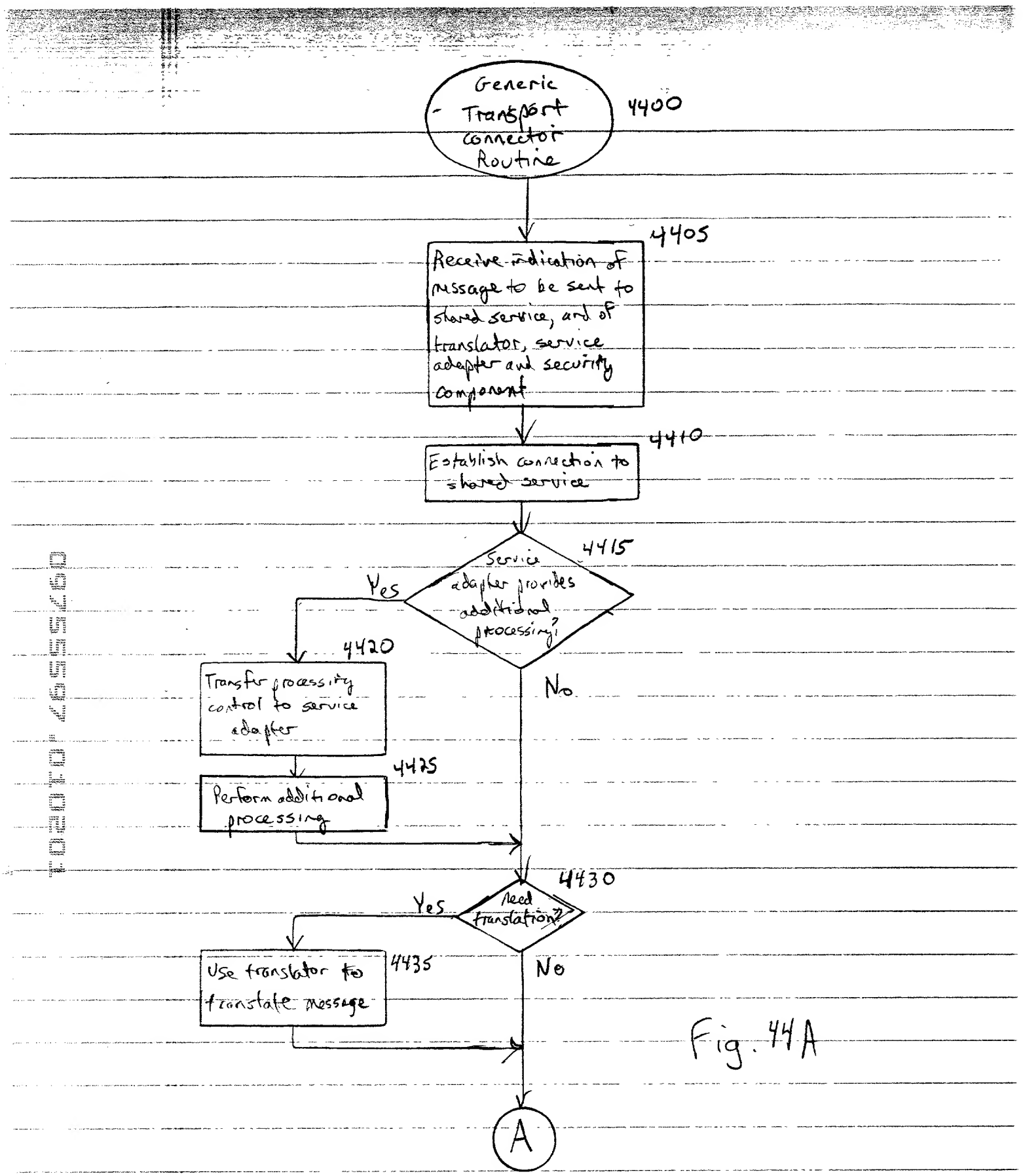

Generic Transport Connector Routine
4400
Receive indication of message to be sent to shared service, and of translator, service adapter and security component
4405
Establish connection to shared service
4410
Service adapter provides additional processing?
4415
Yes
No
Transfer processing control to service adapter
4420
Perform additional processing
4425
Need translation?
4430
Yes
No
Use translator to translate message
4435
A

Fig. 44A

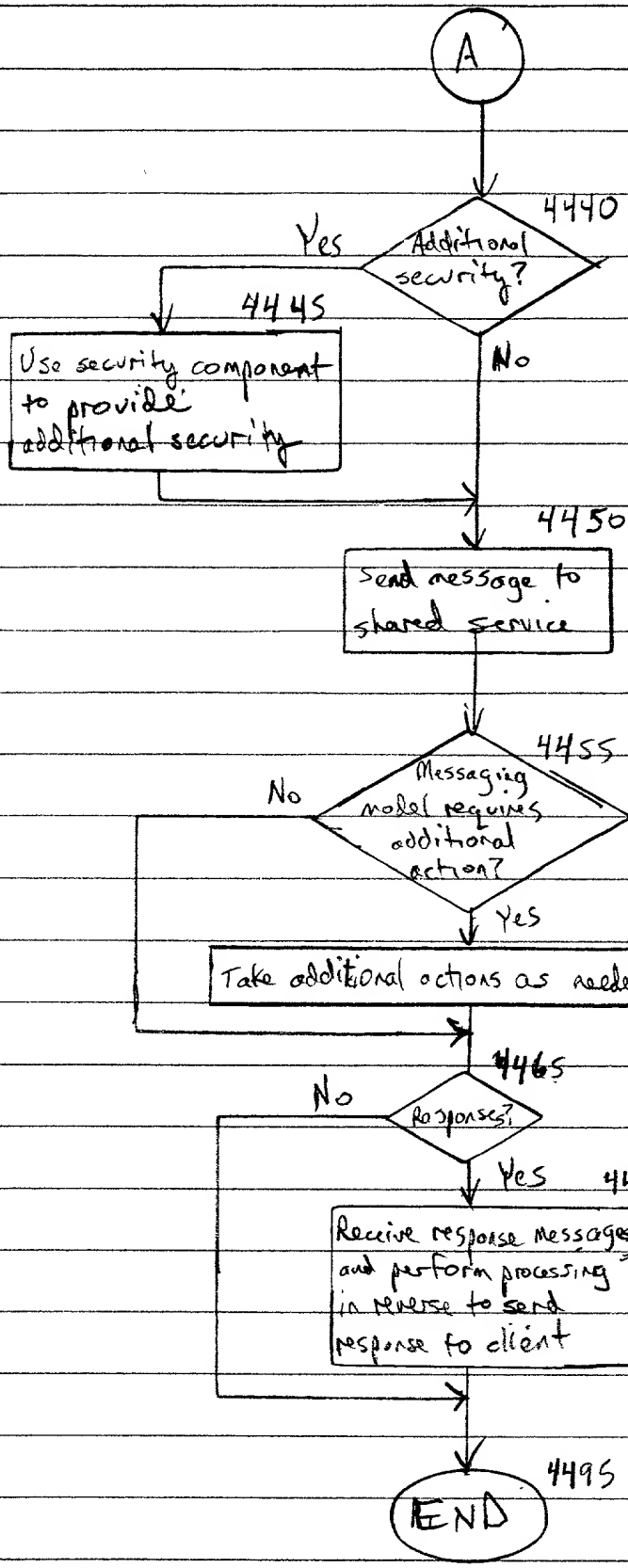

A
4440
Additional security?
Yes
No
4445
Use security component to provide additional security
4450
Send message to shared service
4455
Messaging model requires additional action?
No
Yes
4460
Take additional actions as needed
4465
Responses?
No
Yes
4470
Receive response messages, and perform processing in reverse to send response to client
4495
END

Fig. 44B

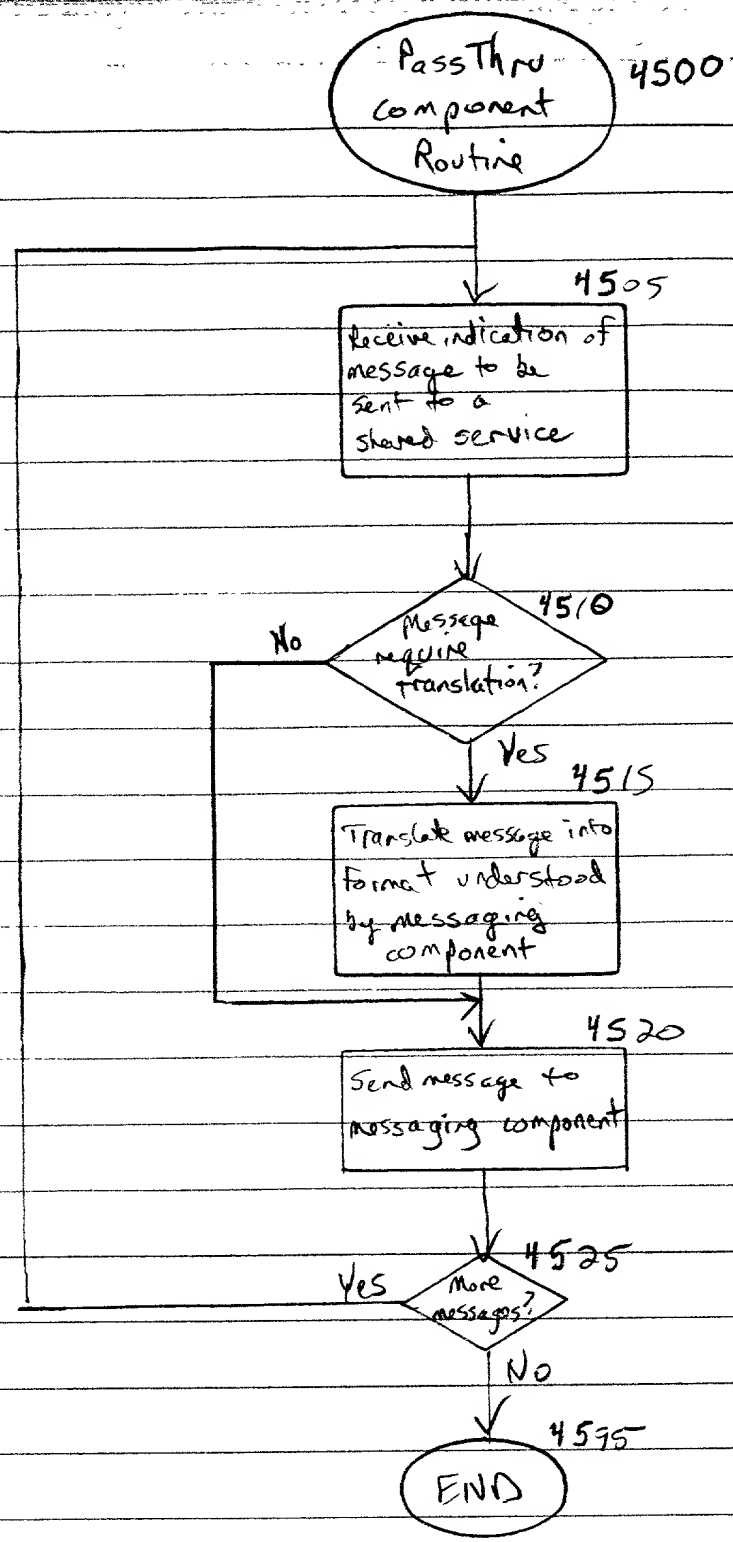

PassThru component Routine
4500
4505
Receive indication of message to be sent to a shared service
4510
Message require translation?
No
Yes
4515
Translate message into format understood by messaging component
4520
Send message to messaging component
4525
More messages?
Yes
No
4595
END

Fig. 45

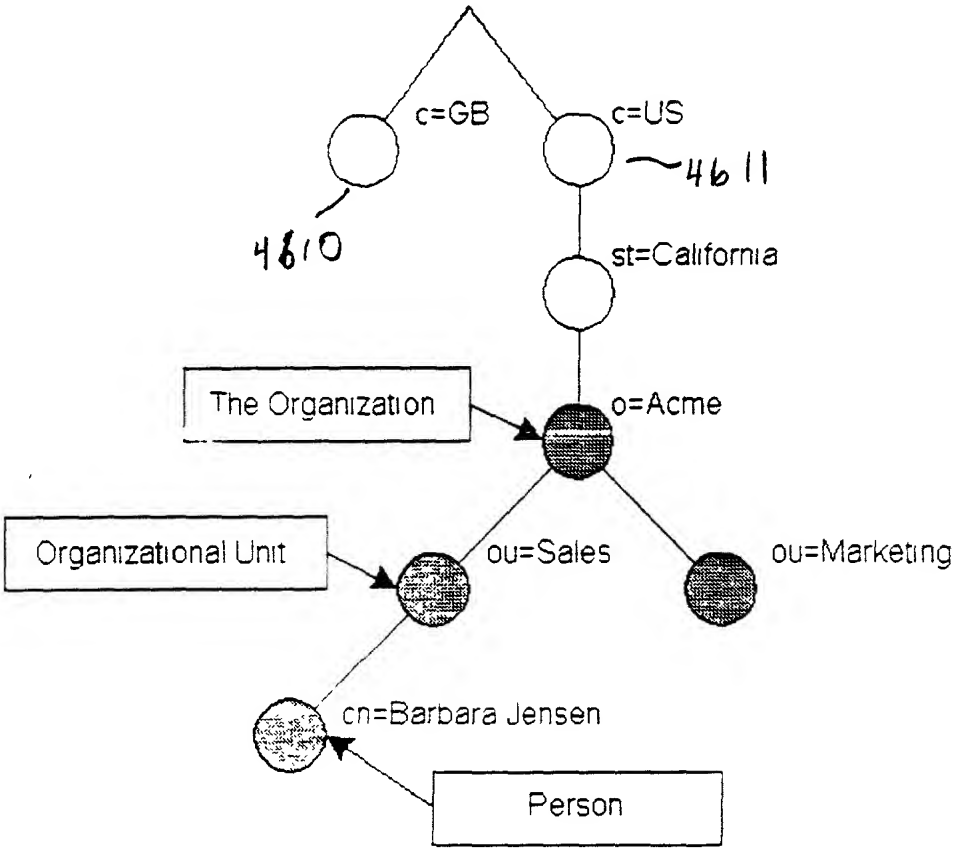


Fig. 46A

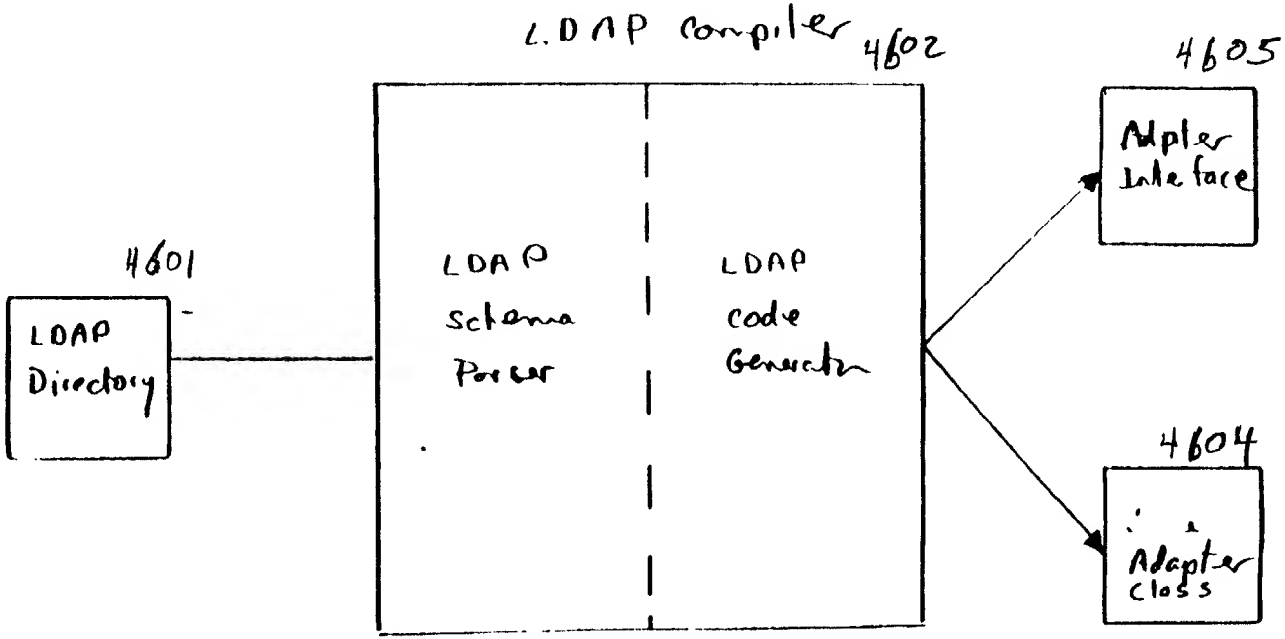

LDAP Compiler
4602
4605
Adpter Interface
4601
LDAP Directory
LDAP Schema Parser
LDAP code Generator
4604
Adapter Class

Fig. 46B

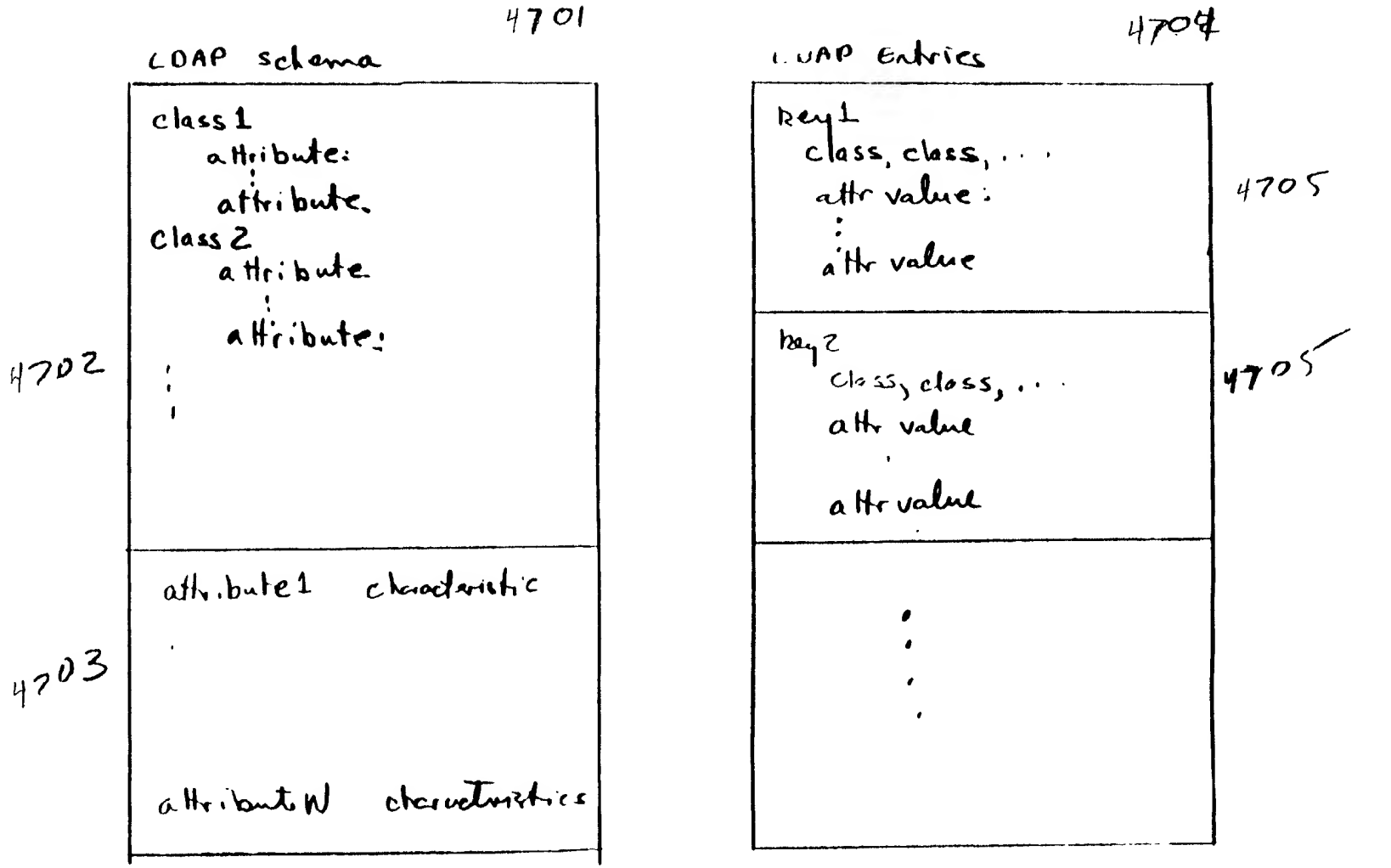

4701
LDAP schema
class 1
attribute:
attribute.
class 2
attribute
attribute:
4702
attribute1 characteristic
4703
attributeN characteristics
4704
LDAP Entries
key1
class, class, ...
attr value:
attr value
4705
key2
class, class, ...
attr value
attr value
4705

Fig 47

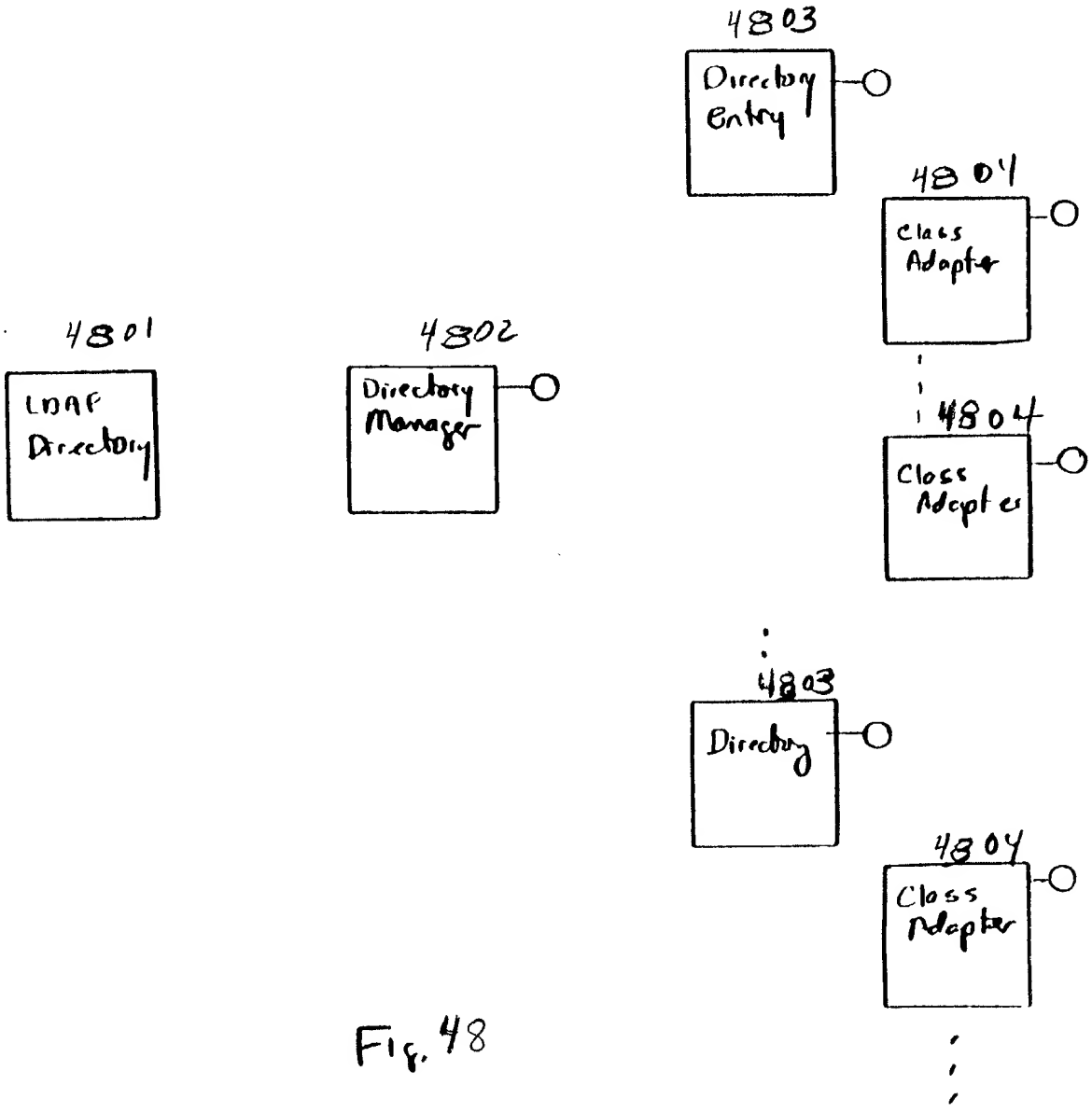

4803
Directory Entry
4804
Class Adapter
4801
LDAP Directory
4802
Directory Manager
4804
Class Adapter
4803
Directory
4804
Class Adapter

Fig. 48

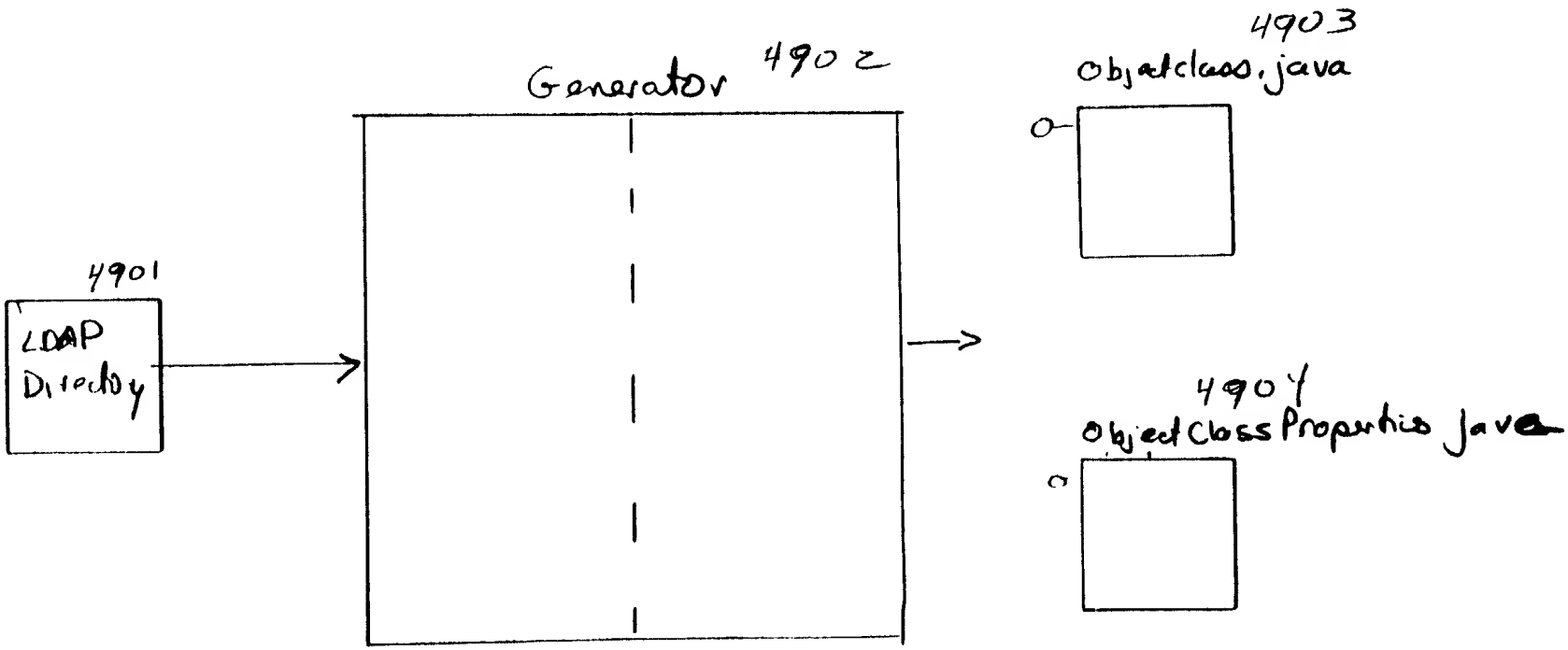

4903
Generator 4902
Objectclass.java
4901
LDAP
Directory
4904
ObjectClassProperties.java

Fig. 49

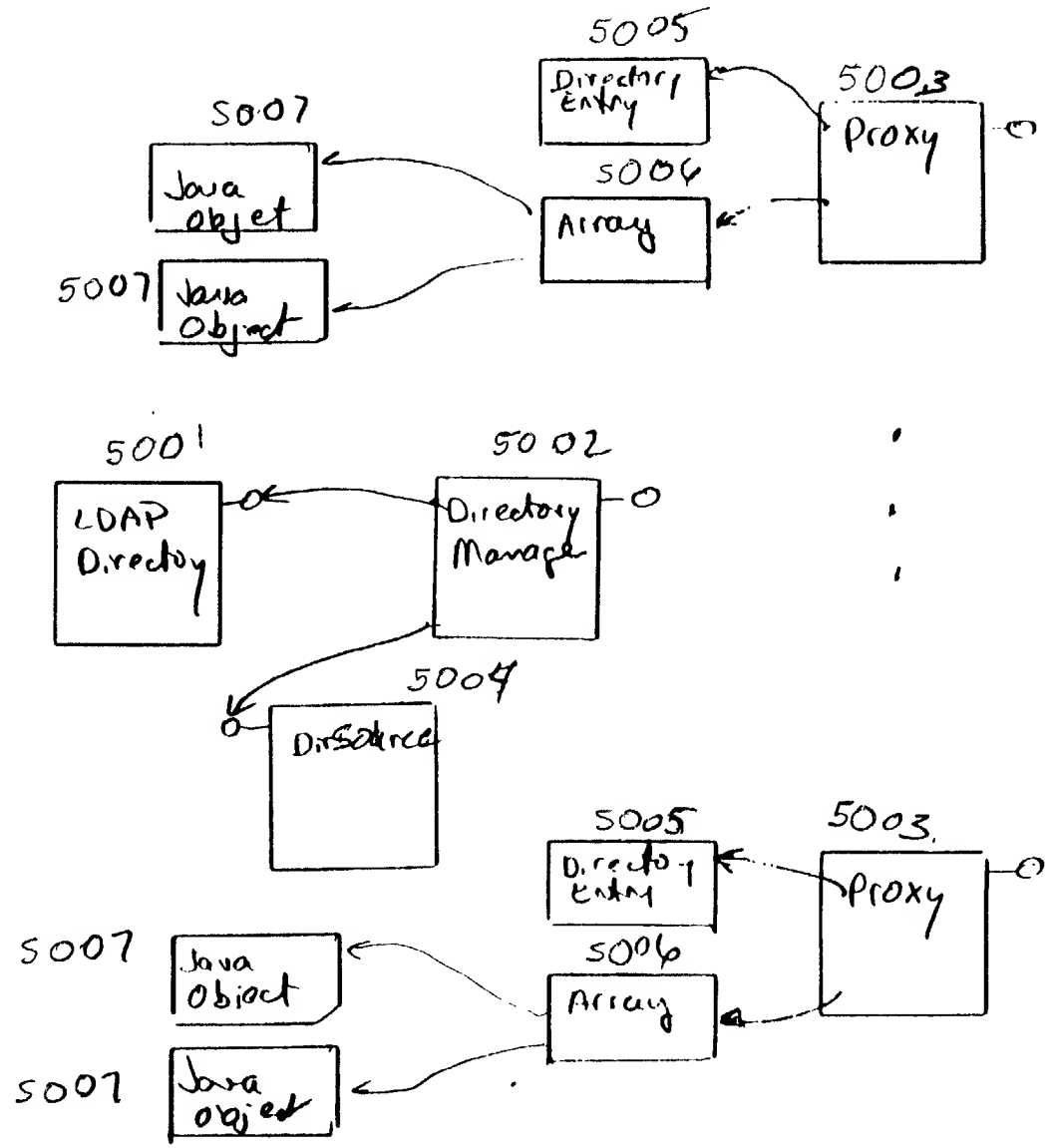

5005
Directory Entry
5003
Proxy
5007
Java object
5006
Array
5007
Java Object
5001
LDAP Directory
5002
Directory Manager
5004
DirSource
5005
Directory Entry
5003
Proxy
5007
Java Object
5006
Array
5007
Java object

Fig. 50

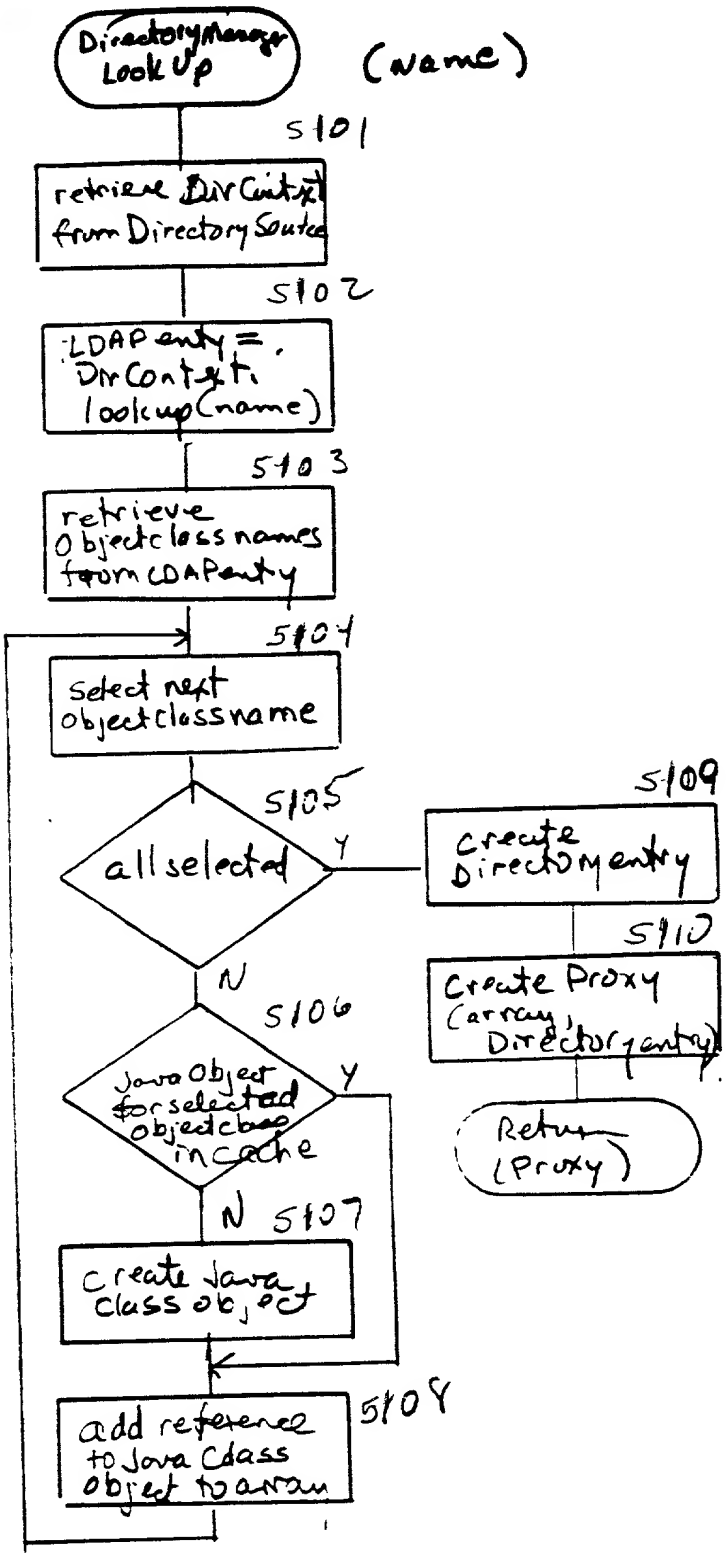

DirectoryManager Look Up
(name)
S101
retrieve DirContext from DirectoryContext
S102
LDAPentry = DirContext. lookup(name)
S103
retrieve Objectclass names from LDAPentry
S104
Select next objectclassname
S105
all selected
Y
N
S106
Java Object for selected objectclass in cache
Y
N
S107
create Java class object
S108
add reference to Java class object to array
S109
create Directory entry
S110
Create Proxy (array, Directory entry)
Return (Proxy)

Fig. 51

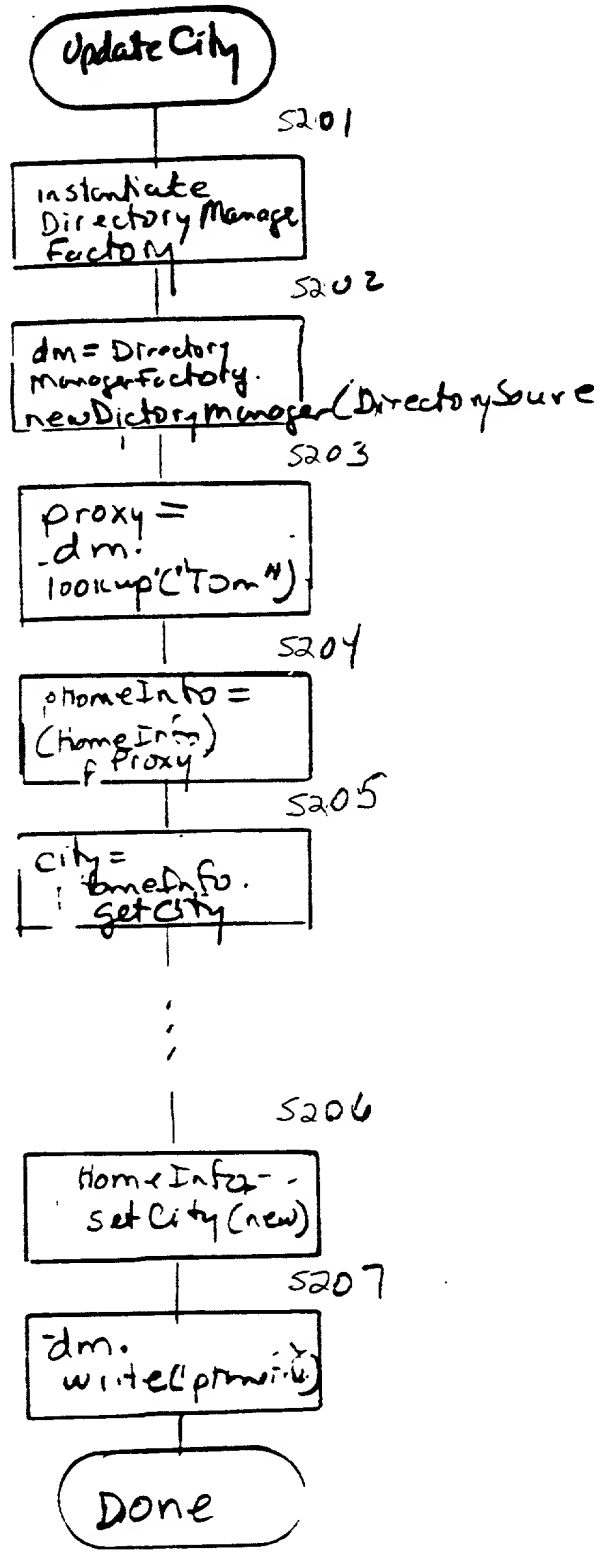


Fig. 52

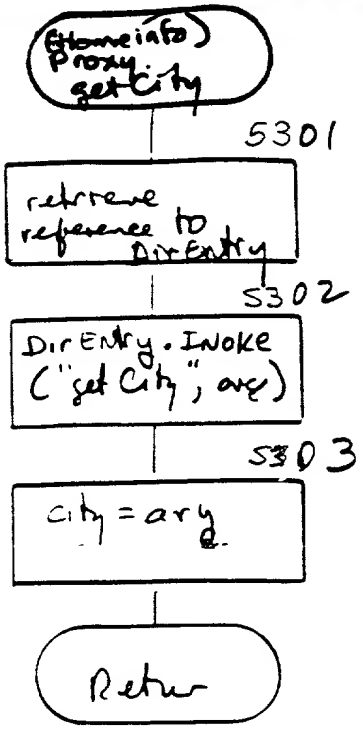
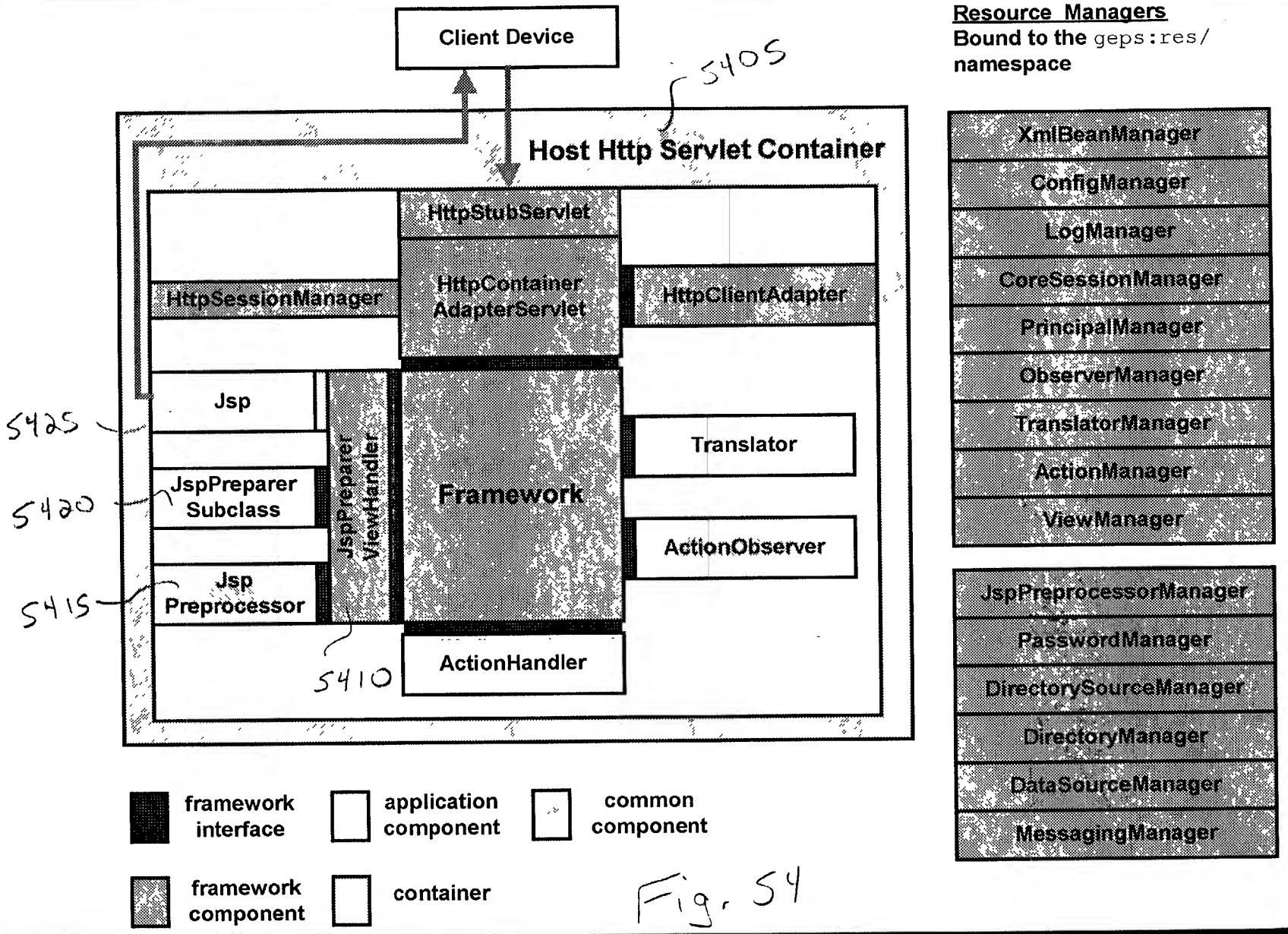


Fig. 53

g GE CASPER Frameworks HTTP Servlet Container



Resource Managers
Bound to the `geps:res/` namespace

- XmlBeanManager
- ConfigManager
- LogManager
- CoreSessionManager
- PrincipalManager
- ObserverManager
- TranslatorManager
- ActionManager
- ViewManager

- JspPreprocessorManager
- PasswordManager
- DirectorySourceManager
- DirectoryManager
- DataSourceManager
- MessagingManager

Fig. 54

com.geps.util.ldap

Class BaseDirectoryAdapter

```
java.lang Object
  |
  +--com.geps.util.ldap.BaseDirectoryAdapter
```

---

public abstract class **BaseDirectoryAdapter**
extends java.lang.Object
implements IBaseObjectClass

```
Class Description:


Base class for all Directory Adapters.  This class is abstract and cannot
be instantiated.
```

Field Summary

protected DirectoryEntry	**m_dirEntry**
protected java.util.ArrayList	**m_modifications**

Constructor Summary

BaseDirectoryAdapter()

Method Summary

DirectoryEntry	**getDirEntry**() Desc: Use to get the DirectoryEntry from the adapter.
java.util.ArrayList	**getModifications**() Desc: Use to get the list of ModificationItem(s) applied to the adapter.
protected void	**initialize**(DirectoryEntry de) Desc: Used to initialize the adapter.

Methods inherited from class java.lang.Object
, clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Field Detail

m_dirEntry

```
protected DirectoryEntry m_dirEntry
```

---

m_modifications

```
protected java.util.ArrayList m_modifications
```

Constructor Detail

BaseDirectoryAdapter

```
public BaseDirectoryAdapter()
```

Method Detail

initialize

```
protected void initialize(DirectoryEntry de)
                  throws javax.naming.NamingException
```

Desc: Used to initialize the adapter. This is used by com.geps.util.ldap.DirectoryManager when its getAdapterInstance() method is called.

Parameters:

de - DirectoryEntry to initialize adapter with.

---

getDirEntry

```
public DirectoryEntry getDirEntry()
```

Desc: Use to get the DirectoryEntry from the adapter.

Specified by:

getDirEntry in interface IBaseObjectClass

Returns:

Returns the DirectoryEntry associated with the adapter.

---

getModifications

```
public java.util.ArrayList getModifications()
```

Desc: Use to get the list of ModificationItem(s) applied to the adapter. This method should not be used by clients. It is used by com.geps.util.ldap.DirectoryManager.

Specified by:

getModifications in interface IBaseObjectClass

Returns:

Returns the list of ModificationItem(s).

---

com.geps.util.ldap

Class DirectoryEntry

```
java.lang.Object
  |
  +--com.geps.util.ldap.DirectoryEntry
```

---

public class **DirectoryEntry**
extends java.lang.Object

Class Description:

Simple wrapper which represents a DirContext.

Field Summary

private java.lang.String	m_dn
private javax.naming.directory.DirContext	m_entry

Constructor Summary

DirectoryEntry(javax.naming.directory.DirContext entry) Desc: Constructor.

Method Summary

javax.naming.directory.DirContext	getDirCtx() Desc: Use to retrieve the entry.
java.lang.String	toString() Desc: Override toString().

Methods inherited from class java.lang.Object

, clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, wait, wait, wait

Field Detail

m_dn

```
private java.lang.String m_dn
```

---

m_entry

```
private javax.naming.directory.DirContext m_entry
```

Constructor Detail

DirectoryEntry

```
public DirectoryEntry(javax.naming.directory.DirContext entry)
```

Desc: Constructor.

Method Detail

getDirCtx

```
public javax.naming.directory.DirContext getDirCtx()
```

Desc: Use to retrieve the entry.
Returns:
Returns the entry.

---

toString

```
public java.lang.String toString()
```

Desc: Override toString().
Overrides:
toString in class java.lang.Object

---

com.geps.util.ldap

Class DirectoryManager

```
java.lang.Object
  |
  +--com.geps.util.ldap.DirectoryManager
```

---

public abstract class **DirectoryManager**
extends java.lang.Object

Class Description:

This class represents the Directory Framework in which clients will interface with to request DirectoryEntry and object class adapters and to write entry modifications back to LDAP.

Field Summary

private static java.lang.String	s_adapterPkg
private static javax.naming.directory.InitialDirContext	s_ctx

Constructor Summary

DirectoryManager()

Method Summary

(package private) static void	()
private static java.lang.String	extractLdapObjClassName(java.lang.String name) Desc: Helper which extracts the object class name from the specified 'name'.
static BaseDirectoryAdapter	getAdapterInstance(DirectoryEntry entry, java.lang.String adapterName) Desc: Use to obtain the specified 'adapterName' adapter from the specified 'entry'.
static java.util.ArrayList	getAdapters(DirectoryEntry entry) Desc: Use to obtain a list of all adapters that the specified 'entry' is composed of.
static DirectoryEntry	getEntry(IBaseObjectClass adapter) Desc: Use to get DirectoryEntry from the specified 'adapter'.
static DirectoryEntry	lookup(java.lang.String dn) Desc: Retrieves the DirectoryEntry whos key matches the the specified 'dn'.
static java.util.ArrayList	search(java.lang.String ctxToSearch, java.lang.String filter) Desc: Use to execute a query against the Directory.

static void	write(IBaseObjectClass adapter) Desc: Use to write out the contents specified by 'adapter' to LDAP.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Field Detail

s_ctx

```
private static javax.naming.directory.InitialDirContext s_ctx
```

---

s_adapterPkg

```
private static java.lang.String s_adapterPkg
```

Constructor Detail

DirectoryManager

```
public DirectoryManager()
```

Method Detail

lookup

```
public static DirectoryEntry lookup(java.lang.String dn)
                             throws javax.naming.NamingException
```

Desc: Retrieves the DirectoryEntry whos key matches the the specified 'dn'.

Returns:

DirectoryEntry associated with the specified 'dn'.

Throws:

javax.naming.NamingException - If a naming exception occurs or if lookup did not return object of type DirContext.

---

getAdapterInstance

```
public static BaseDirectoryAdapter getAdapterInstance(DirectoryEntry entry,
                                                      java.lang.String adapterName)
                                               throws javax.naming.NamingException,
                                                      java.lang.ClassNotFoundException,
                                                      java.lang.InstantiationException,
                                                      java.lang.IllegalAccessException
```

Desc: Use to obtain the specified 'adapterName' adapter from the specified 'entry'. If the requested 'adapterName' is not an object class of 'entry', a null will be returned.

Parameters:

entry - DirectoryEntry in which to search

Returns:
The adapter representing the object class specified by 'adapterName' from the DirectoryEntry 'entry' null is returned if the specified 'adapterName' is not an object class of the specified 'entry'.
Throws:
javax.naming.NamingException - If a naming exception occurs.

---

getEntry

```
public static DirectoryEntry getEntry(IBaseObjectClass adapter)
```

Desc: Use to get DirectoryEntry from the specified 'adapter'.
Parameters:
`adapter` - The adapter to get the DirectoryEntry from.
Returns:
The DirectoryEntry for the specified 'adapter'.

---

write

```
public static void write(IBaseObjectClass adapter)
                  throws javax.naming.NamingException
```

Desc: Use to write out the contents specified by 'adapter' to LDAP.
Parameters:
`adapter` - Object Class to write out.

---

getAdapters

```
public static java.util.ArrayList getAdapters(DirectoryEntry entry)
                                       throws javax.naming.NamingException
```

Desc: Use to obtain a list of all adapters that the specified 'entry' is composed of. Each adapter name returned can be passed into DirectoryManager.getAdapterInstance(DirectoryEntry, String) as the 2nd parameter to obtain an adapter instance.
Parameters:
`entry` - DirectoryEntry in which to discover all adapters for.
Returns:
Non-null ArrayList of String adapters names.

---

search

```
public static java.util.ArrayList search(java.lang.String ctxToSearch,
                                         java.lang.String filter)
                                  throws javax.naming.NamingException
```

Desc: Use to execute a query against the Directory.
Parameters:
`ctxToSearch` - Context to search. "" for current context.
`filter` - LDAP filter.
Returns:
List of DirectoryEntries resulting from the query. Only DirContext objects are supported so if the query returns objects other than DirContext, they will not be in the List.

---

extractLdapObjClassName

```
private static java.lang.String extractLdapObjClassName(java.lang.String name)
```

Desc: Helper which extracts the object class name from the specified 'name'. 'name' looks like "com.geps.ldap.PocuserAdapter". This method removes all package names the "Adapter" suffix is stripped and the remaining string returned.

Parameters:

name - Adapter names which are defined constants in DirectoryConstants or the object class name itself.

Returns:

Returns the LDAP object class name.

```
static void ()
```

---

com.geps.util.ldap

Class Generator

```
java.lang.Object
   |
   +--com.geps.util.ldap.Generator
```

---

public abstract class **Generator**
extends java.lang.Object

Class Description:

This class is used to generate java interfaces and adapters which represents LDAP object classes. For each LDAP object class there is one java interface and one java adapter. This class also generates the file DirectoryConstants which provides defined constants used to identify adapters. These classes \ are used in the java LDAP Directory framework. This class is abstract so it cannot be instantiated.

Inner Class Summary

private static class	Generator.NBP Desc: Helper class, Name Boolean Pair.

Field Summary

private static java.lang.String	s_copyRightYear
private static java.lang.String	s_dateGenerated
private static java.lang.String	s_DirConstName
private static java.lang.String	s_genSrcPath
private static javax.naming.directory.InitialDirContext	s_initDirCtx
private static java.lang.String	s_multiSuffix
private static javax.naming.directory.DirContext	s_schemaRoot
private static java.lang.String	s_srcPathRoot
private static java.lang.String	s_srcPkg
private static java.lang.String	s_ts

Constructor Summary

`Generator()`

Method Summary

(package private) static void	`()`
private static void	`addAttrNameToList(java.util.TreeSet store, javax.naming.NamingEnumeration vals)` Desc: Add attribute names to a TreeSet.
private static void	`emitAdapter(java.lang.String oc, java.util.TreeSet[] attrNames, java.io.PrintStream out)` Desc: Generates the adapter class for the specified object class.
private static void	`emitAdapterClassBody(java.lang.String oc, java.util.TreeSet[] attrNames, java.io.PrintStream out)` Desc: Writes the body of the adapter.
private static void	`emitAdapterGetters(java.util.TreeSet attrSet, java.io.PrintStream out)` Desc: For the specified 'attrSet' will generate getters for all attributes contained within for the adapter.
private static void	`emitAdapterImports(java.io.PrintStream out)` Desc: Writes the import statements for the Adapter.
private static void	`emitAdapterName(java.lang.String oc, java.io.PrintStream out)` Desc: Writes the adapter name and opening curley.
private static void	`emitAdapterSetters(java.util.TreeSet attrSet, java.io.PrintStream out)` Desc: Generate adapter setters for attributes.
private static void	`emitAdapterToString(java.util.TreeSet[] attrNames, java.io.PrintStream out)` Desc: Writes out the adapters toString() method.
private static void	`emitAdapterToStringHelper(java.io.PrintStream out)` Desc: Writes out the adapters toString() helper method.
private static void	`emitAllAdapterGetters(java.util.TreeSet[] attrNames, java.io.PrintStream out)` Desc: Writes out all adapter getters for both required and optional attributes.
private static void	`emitAllAdapterSetters(java.util.TreeSet[] attrNames, java.io.PrintStream out)` Desc: Writes out all adapter setters for both required and optional attributes.
private static void	`emitAllInterfaceGetters(java.util.TreeSet[] attrNames, java.io.PrintStream out)` Desc: Writes out all interface getters for both required and optional attributes.
private static void	`emitAllInterfaceSetters(java.util.TreeSet[] attrNames, java.io.PrintStream out)` Desc: Writes out all interface setters for both required and optional attributes.
private static void	`emitClosingClassBracket(java.io.PrintStream out)` Desc: Writes the class closing curley.
private static void	`emitCommentHeader(java.io.PrintStream out)` Desc: Writes the comment header for the file.
private static void	`emitDirConst(java.lang.String className, java.io.PrintStream out)` Desc: Write out adapter constant for the specified 'className'.
private static void	`emitDirConstName(java.io.PrintStream out)` Desc: Writes the interface name and opening curley.

private static void	**emitGetFromModifiedCache**(java.io.PrintStream out) Desc: Writes out getFromModifiedCache method.
private static void	**emitInterface**(java.lang.String oc, java.util.TreeSet[] attrNames, java.io.PrintStream out) Desc: Generates the interface class for the specified object class.
private static void	**emitInterfaceClassBody**(java.util.TreeSet[] attrNames, java.io.PrintStream out) Desc: Writes the body of the interface.
private static void	**emitInterfaceGetters**(ja[illegible]util.TreeSet attrSet, java.io.PrintStream out) Desc: For the specified 'attrSet' will generate getters for all attributes contained within for the interface.
private static void	**emitInterfaceImp[illegible]rts**(java.io.PrintStream out) Desc: Writes the import statements for the Interface.
private static void	**emitInterfaceName**(java.lang.String oc, java.io.PrintStream out) Desc: Writes the interface name and opening curley.
private static void	**emitInterfaceSetters**(java.util.TreeSet attrSet, java.io.PrintStream out) Desc: For the specified 'attrSet' will generate setters for all attributes contained within for the interface.
private static void	**emitPackage**(java.io.PrintStream out) Desc: Writes the package statement.
private static void	**generate**(java.lang.String[] objClasses) Desc: Directs the generation of interface and adapter classes.
private static void	**getAttributes**(java.lang.String oc, java.util.TreeSet mandatory, java.util.TreeSet optional) Recursively extracts all attributes for the specified 'oc' and all attributes of 'oc' superclasses.
private static java.util.TreeSet[]	**getAttrList**(java.lang.String oc) Desc: Create a list of required and optional attribute names for the specified 'oc' object class and all attributes of 'oc' super object classes and so on by looking up these values in the LDAP Schema.
private static java.lang.String[]	**getObjClasses**() Desc: This method will query LDAP to get all object class names and returns those names in an array of Strings.
private static void	**initialize**() Desc: Gets required system properties, figures out where to create the generated files.
static void	**main**(java.lang.String[] args) Desc: Program entry point.
private static void	**shutDown**() Desc: Release any remaing resources.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Field Detail

s_multiSuffix

```
private static final java.lang.String s_multiSuffix
```

s_ts

```
private static java.lang.String s_ts
```

---

s_DirConstName

```
private static final java.lang.String s_DirConstName
```

---

s_srcPathRoot

```
private static java.lang.String s_srcPathRoot
```

---

s_genSrcPath

```
private static java.lang.String s_genSrcPath
```

---

s_srcPkg

```
private static java.lang.String s_srcPkg
```

---

s_initDirCtx

```
private static javax.naming.directory.InitialDirContext s_initDirCtx
```

---

s_schemaRoot

```
private static javax.naming.directory.DirContext s_schemaRoot
```

---

s_dateGenerated

```
private static java.lang.String s_dateGenerated
```

---

s_copyRightYear

```
private static java.lang.String s_copyRightYear
```

Constructor Detail

Generator

```
public Generator()
```

Method Detail

main

```
public static void main(java.lang.String[] args)
```

Desc: Program entry point. If an object class name does not exist, an error message will be generated and processing continued.

Parameters:

`args` - Array of parameters. If the value of args[0].equals("ALL"), then interfaces/adapters will be generated for all object classes in the LDAP. Any further arguments are ignored. But if the value of args[0] is not equal to "ALL", then the arguments are expected to be LDAP object class names. Each name passed in will be processed and a resulting interface/adapter will be generated.

---

initialize

```
private static void initialize()
                    throws java.lang.Exception
```

Desc: Gets required system properties, figures out where to create the generated files.

---

generate

```
private static void generate(java.lang.String[] objClasses)
                  throws java.lang.Exception
```

Desc: Directs the generation of interface and adapter classes. Interfaces are prefixed with an "I" and adapters are suffixed with "Adapter". Also creates a file which contains constant strings used to identify object classes.

---

emitInterface

```
private static void emitInterface(java.lang.String oc,
                                  java.util.TreeSet[] attrNames,
                                  java.io.PrintStream out)
                           throws javax.naming.NamingException
```

Desc: Generates the interface class for the specified object class.

Parameters:

`oc` - Object Class to generate interface for.

`attrNames` - Array of TreeSet object containing the required and optional attribute names. Required is at index 0, optional at index 1.

`out` - Stream to write to.

---

emitAdapter

```
private static void emitAdapter(java.lang.String oc,
                                java.util.TreeSet[] attrNames,
                                java.io.PrintStream out)
                         throws javax.naming.NamingException
```

Desc: Generates the adapter class for the specified object class.
Parameters:
`oc` - Object Class to generate adapter for.
`attrNames` - Array of TreeSet object containing the required and optional attribute names. Required is at index 0, optional at index 1.
`out` - Stream to write to.

---

emitCommentHeader

```
private static void emitCommentHeader(j  a.io.PrintStream out)
```

Desc: Writes the comment header for the file.
Parameters:
`out` - Stream to write to.

---

emitPackage

```
private static void emitPackage(java.io.PrintStream out)
```

Desc: Writes the package statement.
Parameters:
`out` - Stream to write to.

---

emitInterfaceImports

```
private static void emitInterfaceImports(java.io.PrintStream out)
```

Desc: Writes the import statements for the Interface.
Parameters:
`out` - Stream to write to.

---

emitAdapterImports

```
private static void emitAdapterImports(java.io.PrintStream out)
```

Desc: Writes the import statements for the Adapter.
Parameters:
`out` - Stream to write to.

---

emitInterfaceName

```
private static void emitInterfaceName(java.lang.String oc,
                                      java.io.PrintStream out)
```

Desc: Writes the interface name and opening curley.
Parameters:
`oc` - Object class name.
`out` - Stream to write to.

---

emitAdapterName

```
private static void emitAdapterName(java.lang.String oc,
                                    java.io.PrintStream out)
```

Desc: Writes the adapter name and opening curley.
Parameters:
oc - Object class name.
out - Stream to write to.

---

emitDirConstName

```
private static void emitDirConstName(java.io.PrintStream out)
```

Desc: Writes the interface name and opening curley.
Parameters:
oc - Object class name.
out - Stream to write to.

---

emitInterfaceClassBody

```
private static void emitInterfaceClassBody(java.util.TreeSet[] attrNames,
                                           java.io.PrintStream out)
                                    throws javax.naming.NamingException
```

Desc: Writes the body of the interface.
Parameters:
attrNames - An array of TreeSet objects containing the required and optional object class attribute names.
TreeSet[0] = requried, TreeSet[1] = optional.
out - Stream to write to.

---

emitAdapterClassBody

```
private static void emitAdapterClassBody(java.lang.String oc,
                                         java.util.TreeSet[] attrNames,
                                         java.io.PrintStream out)
                                  throws javax.naming.NamingException
```

Desc: Writes the body of the adapter.
Parameters:
oc - Object class name.
attrNames - An array of TreeSet objects containing the required and optional object class attribute names.
TreeSet[0] = requried, TreeSet[1] = optional.
out - Stream to write to.

---

emitAllInterfaceGetters

```
private static void emitAllInterfaceGetters(java.util.TreeSet[] attrNames,
                                            java.io.PrintStream out)
```

Desc: Writes out all interface getters for both required and optional attributes.
Parameters:
... An array of TreeSet containing the required and optional attribute names.

out - Stream to write to.

---

emitAllInterfaceSetters

```
private static void emitAllInterfaceSetters(java.util.TreeSet[] attrNames,
                                            java.io.PrintStream out)
```

Desc: Writes out all interface setters for both required and optional attributes.
Parameters:
attrNames - An array of TreeSet containing the required and optional attribute names.
out - Stream to write to.

---

emitAllAdapterGetters

```
private static void emitAllAdapterGetters(java.util.TreeSet[] attrNames,
                                          java.io.PrintStream out)
```

Desc: Writes out all adapter getters for both required and optional attributes.
Parameters:
attrNames - An array of TreeSet containing the required and optional attribute names.
out - Stream to write to.

---

emitAllAdapterSetters

```
private static void emitAllAdapterSetters(java.util.TreeSet[] attrNames,
                                          java.io.PrintStream out)
```

Desc: Writes out all adapter setters for both required and optional attributes.
Parameters:
attrNames - An array of TreeSet containing the required and optional attribute names.
out - Stream to write to.

---

emitInterfaceGetters

```
private static void emitInterfaceGetters(java.util.TreeSet attrSet,
                                         java.io.PrintStream out)
```

Desc: For the specified 'attrSet' will generate getters for all attributes contained within for the interface.
Parameters:
attrSet - Set of attribute names to generate getters for.
out - Stream to write to.

---

emitInterfaceSetters

```
private static void emitInterfaceSetters(java.util.TreeSet attrSet,
                                         java.io.PrintStream out)
```

Desc: For the specified 'attrSet' will generate setters for all attributes contained within for the interface.
Parameters:
attrSet - Set of attribute names to generate setters for.
out - Stream to write to.

---

emitAdapterSetters

```
private static void emitAdapterSetters(java.util.TreeSet attrSet,
                                       java.io.PrintStream out)
```

Desc: Generate adapter setters for attributes.
Parameters:
`attrSet` - Set of attribute names to generate setters for.
`out` - Stream to write to.

---

emitAdapterGetters

```
private static void emitAdapterGetters(java.util.TreeSet attrSet,
                                       java.io.PrintStream out)
```

Desc: For the specified 'attrSet' will generate getters for all attributes contained within for the adapter.
Parameters:
`attrSet` - Set of attribute names to generate getters for.
`out` - Stream to write to.

---

emitGetFromModifiedCache

```
private static void emitGetFromModifiedCache(java.io.PrintStream out)
```

Desc: Writes out getFromModifiedCache method. This is a getter helper method which looks into the modified cache for changes.
Parameters:
`out` - Stream to write to.

---

emitAdapterToString

```
private static void emitAdapterToString(java.util.TreeSet[] attrNames,
                                        java.io.PrintStream out)
```

Desc: Writes out the adapters toString() method.
Parameters:
`attrNames` - An array of TreeSet containing the required and optional attribute names.
`out` - Stream to write to.

---

emitAdapterToStringHelper

```
private static void emitAdapterToStringHelper(java.io.PrintStream out)
```

Desc: Writes out the adapters toString() helper method.
Parameters:
`out` - Stream to write to.

---

emitDirConst

```
private static void emitDirConst(java.lang.String className,
                                 java.io.PrintStream out)
```

Desc: Write out adapter constant for the specified 'className'.
Parameters:
`className` - Object class name to write constant for.
`out` - Stream to write to.

---

emitClosingClassBracket

```
private static void emitClosingClassBracket(java.io.PrintStream out)
```

Desc: Writes the class closing curley.
Parameters:
`out` - Stream to write to.

---

shutDown

```
private static void shutDown()
                    throws javax.naming.NamingException
```

Desc: Release any remaing resources.

---

getObjClasses

```
private static java.lang.String[] getObjClasses()
                                        throws javax.naming.NamingException
```

Desc: This method will query LDAP to get all object class names and returns those names in an array of Strings.
Returns:
Returns an array of Strings containing all object class names.
Throws:
javax.naming.NamingException - If a naming exception occurs.

---

getAttrList

```
private static java.util.TreeSet[] getAttrList(java.lang.String oc)
                                      throws javax.naming.NamingException
```

Desc: Create a list of required and optional attribute names for the specified 'oc' object class and all attributes of 'oc' super object classes and so on by looking up these values in the LDAP Schema. Along with each attribute name is a boolean flag which indicates if the attribute is single valued or not. Returns this information in an array of TreeSet objects. The first element in the array contains the required attributes and the second element contains the optional attributes. The elements in contained in the TreeSet are Generator.NBP objects (Name Boolean Pair). The name is the attribute name and the boolean indicates if it is single valued or not.
Parameters:
`oc` - Object class name to build attribute list for.
Returns:
Array of TreeSet containing the required and optional attributes.

---

getAttributes

```
private static void getAttributes(java.lang.String oc,
                                  java.util.TreeSet mandatory,
                                  java.util.TreeSet optional)
                           throws javax.naming.NamingException
```

Recursively extracts all attributes for the specified 'oc' and all attributes of 'oc' superclasses.

Parameters:

`oc` - Object class name in which to extract attributes for.
`mandatory` - TreeSet to store mandatory attributes.
`optional` - TreeSet to store optional attributes.

---

addAttrNameToList

```
private static void addAttrNameToList(java.util.TreeSet store,
                                      javax.naming.NamingEnumeration vals)
                               throws javax.naming.NamingException
```

Desc: Add attribute names to a TreeSet. TreeSet does not allow dups. For each attribute 'vals', this method also determines if that attribute is single or multivalued. The attribute name and whether it is single valued or not is added to 'store' as Generator.NBP (Name Boolean Pair) object.

Parameters:

`store` - TreeSet to store attribute names
`vals` - Enumeration of attribute names

```
static void ()
```

---

com.geps.util.ldap

Class Generator.NBP

```
java.lang.Object
  |
  +--com.geps.util.ldap.Generator.NBP
```

Enclosing class:
Generator

---

private static class **Generator.NBP**
extends java.lang.Object
implements java.lang.Comparable

Desc: Helper class, Name Boolean Pair. Holds attributeName, isSingle boolean pair.

Field Summary

java.lang.String	**m_attrName**
boolean	**m_isSingle**

Constructor Summary

Generator.NBP(java.lang.String attrName, boolean isSingle)

Method Summary

int	**compareTo**(java.lang.Object obj)

Methods inherited from class java.lang.Object

, clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Field Detail

m_attrName

public java.lang.String **m_attrName**

---

m_isSingle

```
public boolean m_isSingle
```

Constructor Detail

Generator.NBP

```
public Generator.NBP(java.lang.String attrName,
                     boolean isSingle)
```

Method Detail

compareTo

```
public int compareTo(java.lang.Object obj)
```

Specified by:
compareTo in interface java.lang.Comparable

com.geps.util.ldap

Interface IBaseObjectClass

All Known Implementing Classes:
BaseDirectoryAdapter

---

public interface **IBaseObjectClass**

Class Description:

Base interface for all object class interfaces.

Method Summary

DirectoryEntry	**getDirEntry**()
java.util.ArrayList	**getModifications**()

Method Detail

getModifications

public java.util.ArrayList **getModifications**()

---

getDirEntry

public DirectoryEntry **getDirEntry**()

---

com.geps.util.ldap

Class Util

```
java.lang.Object
  |
  +--com.geps.util.ldap.Util
```

---

public abstract class **Util**
extends java.lang.Object

Class Description:

Utility methods used by classes in the com.geps.util.ldap package. This class is abstract and cannot be instantiated.

Field Summary

static java.lang.String	**ADAPTER_SUFFIX**
static java.lang.String	**INTERFACE_PREFIX**

Constructor Summary

Util()

Method Summary

static java.lang.String	**convertToValidMethodName**(java.lang.String str Desc: Will return a string with the same contents of 'str' out with the 1st character uppercased, the rest of the characters lower case and any '-' to '_'.

Methods inherited from class java.lang.Object
, clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Field Detail

INTERFACE_PREFIX

```
public static final java.lang.String INTERFACE_PREFIX
```

Appendix B

java.lang.reflect

Class Proxy

```
java.lang.Object
  |
  +--java.lang.reflect.Proxy
```

All Implemented Interfaces:
Serializable

---

public class **Proxy**
extends Object
implements Serializable

`Proxy` provides static methods for creating dynamic proxy classes and instances, and it is also the superclass of all dynamic proxy classes created by those methods

To create a proxy for some interface `Foo`

```
InvocationHandler handler = new MyInvocationHandler(...);
Class proxyClass = Proxy.getProxyClass(
    Foo.class.getClassLoader(), new Class[] { Foo.class });
Foo f = (Foo) proxyClass.
    getConstructor(new Class[] { InvocationHandler.class }).
    newInstance(new Object[] { handler });
```

or more simply

```
Foo f = (Foo) Proxy.newProxyInstance(Foo.class.getClassLoader(),
                                     new Class[] { Foo.class },
                                     handler);
```

A *dynamic proxy class* (simply referred to as a *proxy class* below) is a class that implements a list of interfaces specified at runtime when the class is created, with behavior as described below. A *proxy interface* is such an interface that is implemented by a proxy class. A *proxy instance* is an instance of a proxy class. Each proxy instance has an associated *invocation handler* object, which implements the interface `InvocationHandler`. A method invocation on a proxy instance through one of its proxy interfaces will be dispatched to the `invoke` method of the instance's invocation handler, passing the proxy instance, a `java.lang.reflect.Method` object identifying the method that was invoked, and an array of type `Object` containing the arguments. The invocation handler processes the encoded method invocation as appropriate and the result that it returns will be returned as the result of the

method invocation on the proxy instance.

A proxy class has the following properties.

- Proxy classes are public, final, and not abstract
- The unqualified name of a proxy class is unspecified The space of class names that begin with the string `"$Proxy"` should be, however, reserved for proxy classes
- A proxy class extends `java.lang.reflect.Proxy`.
- A proxy class implements exactly the interfaces specified at its creation, in the same order
- If a proxy class implements a non-public interface, then it will be defined in the same package as that interface Otherwise, the package of a proxy class is also unspecified. Note that package sealing will not prevent a proxy class from being successfully defined in a particular package at runtime, and neither will classes already defined in the same class loader and the same package with particular signers.
- Since a proxy class implements all of the interfaces specified at its creation, invoking `getInterfaces` on its `Class` object will return an array containing the same list of interfaces (in the order specified at its creation), invoking `getMethods` on its `Class` object will return an array of `Method` objects that include all of the methods in those interfaces, and invoking `getMethod` will find methods in the proxy interfaces as would be expected
- The `Proxy.isProxyClass` method will return true if it is passed a proxy class-- a class returned by `Proxy.getProxyClass` or the class of an object returned by `Proxy.newProxyInstance`-- and false otherwise
- The `java.security.ProtectionDomain` of a proxy class is the same as that of system classes loaded by the bootstrap class loader, such as `java.lang.Object`, because the code for a proxy class is generated by trusted system code This protection domain will typically be granted `java.security.AllPermission`
- Each proxy class has one public constructor that takes one argument, an implementation of the interface `InvocationHandler`, to set the invocation handler for a proxy instance. Rather than having to use the reflection API to access the public constructor, a proxy instance can be also be created by calling the `Proxy.newInstance` method, which combines the actions of cal' :g `Proxy.getProxyClass` with invoking the constructor with an invocation handler

A proxy instance has the following properties

- Given a proxy instance `proxy` and one of the interfaces implemented by its proxy class `Foo`, the following expression will return true.

```
proxy instanceof Foo
```

and the following cast operation will succeed (rather than throwing a `ClassCastException`)

```
(Foo) proxy
```

- Each proxy instance has an associated invocation handler, the one that was passed to its constructor The static `Proxy.getInvocationHandler` method will return the invocation handler associated with the proxy instance passed as its argument.
- An interface method invocation on a proxy instance will be encoded and dispatched to the invocation handler's `invoke` method as described in the documentation for that method.
- An invocation of the `hashCode`, `equals`, or `toString` methods declared in `java.lang.Object` on a proxy

instance will be encoded and dispatched to the invocation handler's `invoke` method in the same manner as interface method invocations are encoded and dispatched, as described above. The declaring class of the `Method` object passed to `invoke` will be `java.lang.Object` Other public methods of a proxy instance inherited from `java.lang.Object` are not overridden by a proxy class, so invocations of those methods behave like they do for instances of `java.lang.Object`.

Methods Duplicated in Multiple Proxy Interfaces

When two or more interfaces of a proxy class contain a method with the same name and parameter signature, the order of the proxy class's interfaces becomes significant When such a *duplicate method* is invoked on a proxy instance, the `Method` object passed to the invocation handler will not necessarily be the one whose declaring class is assignable from the reference type of the interface that the proxy's method was invoked through This limitation exists because the corresponding method implementation in the generated proxy class cannot determine which interface it was invoked through Therefore, when a duplicate method is invoked on a proxy instance, the `Method` object for the method in the foremost interface that contains the method (either directly or inherited through a superinterface) in the proxy class's list of interfaces is passed to the invocation handler's `invoke` method, regardless of the reference type through which the method invocation occurred

If a proxy interface contains a method with the same name and parameter signature as the `hashCode`, `equals`, or `toString` methods of `java.lang.Object`, when such a method is invoked on a proxy instance, the `Method` object passed to the invocation handler will have `java.lang.Object` as its declaring class. In other words, the public, non-final methods of `java.lang.Object` logically precede all of the proxy interfaces for the determination of which `Method` object to pass to the invocation handler

Note also that when a duplicate method is dispatched to an invocation handler, the `invoke` method may only throw checked exception types that are assignable to one of the exception types in the `throws` clause of the method in *all* of the proxy interfaces that it can be invoked through If the `invoke` method throws a checked exception that is not assignable to any of the exception types declared by the method in one of the the proxy interfaces that it can be invoked through, then an unchecked `UndeclaredThrowableException` will be thrown by the invocation on the proxy instance This restriction means that not all of the exception types returned by invoking `getExceptionTypes` on the `Method` object passed to the `invoke` method can necessarily be thrown successfully by the `invoke` method

Since:

JDK1.3

See Also:

`InvocationHandler`, Serialized Form

Field Summary

protected InvocationHandler	**h** the invocation handler for this proxy instance

Constructor Summary

protected	**Proxy**(InvocationHandler h) Constructs a new Proxy instance from a subclass (typically, a dynamic proxy class) with the specified value for its invocation handler

Method Summary

static InvocationHandler	**getInvocationHandler**(Object proxy) Returns the invocation handler for the specified proxy instance
static Class	**getProxyClass**(ClassLoader loader, Class[] interfaces) Returns the java.lang.Class object for a proxy class given a class loader and an array of interfaces
static boolean	**isProxyClass**(Class cl) Returns true if and only if the specified class was dynamically generated to be a proxy class using the getProxyClass method or the newProxyInstance method
static Object	**newProxyInstance**(ClassLoader loader, Class[] interfaces, InvocationHandler h) Returns an instance of a proxy class for the specified interfaces that dispatches method invocations to the specified invocation handler.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

h

```
protected InvocationHandler h
```

the invocation handler for this proxy instance

Constructor Detail

Proxy

```
protected Proxy(InvocationHandler h)
```

Constructs a new Proxy instance from a subclass (typically, a dynamic proxy class) with the specified value for its invocation handler

Parameters:

h - the invocation handler for this proxy instance

Method Detail

getProxyClass

```
public static Class getProxyClass(ClassLoader loader,
                                   Class[] interfaces)
                            throws IllegalArgumentException
```

Returns the java.lang.Class object for a proxy class given a class loader and an array of interfaces The proxy class will be defined in the specified class loader and will implement all of the supplied interfaces If a proxy class for the same permutation of interfaces has already been defined in the class loader, then the existing proxy class will be returned, otherwise, a proxy class for those interfaces will be generated dynamically and defined in the class loader

There are several restrictions on the parameters that may be passed to Proxy.getProxyClass.

- All of the Class objects in the interfaces array must represent interfaces, not classes or primitive types
- No two elements in the interfaces array may refer to identical Class objects
- All of the interface types must be visible by name through the specified class loader In other words, for class loader cl and every interface i, the following expression must be true

```
Class.forName(i.getName(), false, cl) == i
```

- All non-public interfaces must be in the same package; otherwise, it would not be possible for the proxy class to implement all of the interfaces, regardless of what package it is defined in
- No two interfaces may each have a method with the same name and parameter signature but different return type
- The resulting proxy class must not exceed any limits imposed on classes by the virtual machine For example, the VM may limit the number of interfaces that a class may implement to 65535; in that case, the size of the interfaces array must not exceed 65535

If any of these restrictions are violated, Proxy.getProxyClass will throw an IllegalArgumentException. If the interfaces array argument or any of its elements are null, a NullPointerException will be thrown

Note that the order of the specified proxy interfaces is significant. two requests for a proxy class with the same combination of interfaces but in a different order will result in two distinct proxy classes.

Parameters:

loader - the class loader to define the proxy class in
interfaces - the list of interfaces for the proxy class to implement

Returns:

a proxy class that is defined in the specified class loader and that implements the specified interfaces

Throws:

IllegalArgumentException - if any of the restrictions on the parameters that may be passed to getProxyClass are violated
NullPointerException - if the interfaces array argument or any of its elements are null

---

newProxyInstance

```
public static Object newProxyInstance(ClassLoader loader,
                                       Class[] interfaces,
                                       InvocationHandler h)
                                throws IllegalArgumentException
```

Returns an instance of a proxy class for the specified interfaces that dispatches method invocations to the specified invocation handler. This method is equivalent to:

```
Proxy.getProxyClass(loader, interfaces).
    getConstructor(new Class[] { InvocationHandler.class }).
    newInstance(new Object[] { handler });
```

Proxy.newProxyInstance throws IllegalArgumentException for the same reasons that Proxy.getProxyClass does

Parameters:

loader - the class loader to define the proxy class in
interfaces - the list of interfaces for the proxy class to implement
h - the invocation handler to dispatch method invocations to

Returns:

a proxy instance with the specified invocation handler of a proxy class that is defined in the specified class loader and that implements the specified interfaces

Throws:

IllegalArgumentException - if any of the restrictions on the parameters that may be passed to getProxyClass are violated
NullPointerException - if the interfaces array argument or any of its elements are null, or if the invocation handler, h, is null

---

isProxyClass

```
public static boolean isProxyClass(Class cl)
```

Returns true if and only if the specified class was dynamically generated to be a proxy class using the getProxyClass method or the newProxyInstance method.

The reliability of this method is important for the ability to use it to make security decisions, so its implementation should not just test if the class in question extends Proxy

Parameters:

`cl` - the class to test

Returns:

`true` if the class is a proxy class and `false` otherwise

Throws:

`NullPointerException` - if `cl` is `null`

---

getInvocationHandler

```
public static InvocationHandler getInvocationHandler(Object proxy)
                                                throws IllegalArgumentException
```

Returns the invocation handler for the specified proxy instance.

Parameters:

`proxy` - the proxy instance to return the invocation handler for

Returns:

the invocation handler for the proxy instance

Throws:

`IllegalArgumentException` - if the argument is not a proxy instance

---

java.lang.reflect

Interface InvocationHandler

public interface **InvocationHandler**

`InvocationHandler` is the interface implemented by the *invocation handler* of a proxy instance

Each proxy instance has an associated invocation handler When a method is invoked on a proxy instance, the method invocation is encoded and dispatched to the `invoke` method of its invocation handler

Since:

JDK1.3

See Also:

Proxy

Method Summary

Object	**invoke**(Object proxy, Method method, Object[] args) Processes a method invocation on a proxy instance and returns the result

Method Detail

invoke

```
public Object invoke(Object proxy,
                     Method method,
                     Object[] args)
              throws Throwable
```

Processes a method invocation on a proxy instance and returns the result. This method will be invoked on an invocation handler when a method is invoked on a proxy instance that it is associated with.

Parameters:

`proxy` - the proxy instance that the method was invoked on

`method` - the `Method` instance corresponding to the interface method invoked on the proxy instance The declaring class of the `Method` object will be the interface that the method was declared in, which may be a superinterface of the proxy interface that the proxy class inherits the method through

`args` - an array of objects containing the values of the arguments passed in the method invocation on the proxy instance, or `null` if interface method takes no arguments Arguments of primitive types are wrapped in instances of the appropriate primitive wrapper class, such as `java.lang.Integer` or `java.lang.Boolean`

Returns:

the value to return from the method invocation on the proxy instance If the declared return type of the interface method is a primitive type, then the value returned by this method must be an instance of the corresponding primitive wrapper class, otherwise, it must be a type assignable to the declared return type If the value returned by this method is `null` and the interface method's return type is primitive, then a `NullPointerException` will be thrown by the method invocation on the proxy instance If the value returned by this method is otherwise not compatible with the interface method's declared return type as described above, a `ClassCastException` will be thrown by the method invocation on the proxy instance

Throws:

`Throwable` - the exception to throw from the method invocation on the proxy instance The exception's type must be assignable either to any of the exception types declared in the `throws` clause of the interface method or to the unchecked exception types `java.lang.RuntimeException` or `java.lang.Error`. If a checked exception is thrown by this method that is not assignable to any of the exception types declared in the `throws` clause of the interface method, then an `UndeclaredThrowableException` containing the exception that was thrown by this method will be thrown by the method invocation on the proxy instance

See Also:

`UndeclaredThrowableException`

Appendix C

Package com.gepower.sfo.tool.ldap

Interface Summary	
DirectoryManager	DirectoryManager provides an interface for accessing Directory data
DirectorySource	DirectorySource is an interface which provides access to Directory data sources

Class Summary	
DefaultDirectorySource	DirectorySource implementation
DirectoryEntry	Represents an LDAP Directory Entry and a LDAP invocation handler used in Proxy instances
DirectoryManagerFactory	Use to create an object which implements the DirectoryManager interface
Generator	Generates java interfaces which represents LDAP object classes.

Package com.gepower.sfo.tool.ldap

Interface Summary	
DirectoryManager	DirectoryManager provides an interface for accessing Directory data
DirectorySource	DirectorySource is an interface which provides access to Directory data sources

Class Summary	
DefaultDirectorySource	DirectorySource implementation.
DirectoryEntry	Represents an LDAP Directory Entry and a LDAP invocation handler used in Proxy instances.
DirectoryManagerFactory	Use to create an object which implements the DirectoryManager interface
Generator	Generates java interfaces which represents LDAP object classes.

com.gepower.sfo.tool.ldap

Interface DirectoryManager

public interface **DirectoryManager**

Dire ryManager provides an interface for accessing Directory data

Method Summary

java.lang.[illegible]	**cast**(java.lang.Object entry, java.lang.Class interfaceToCastTo) This method provides backward compatibility from Java 1.3 to Java 1.2 which does not support the Proxy class.
java.lang.Str[illegible]	**getDN**(java.lang.Object entry) Use to get the specified 'entry' distinquished name
java.lang.Object	**lookup**(java.lang.String dn) Use to lookup a specific entry identified by the specified 'dn'
java.lang.[illegible]	**mixinInterfaces**(java.lang.Object entry, java.lang.Class[] newInterfaces) Use to add additional interfaces specified by 'newInterfaces' to an existing 'entry'
java.lang.Object	**newEntryInstance**(java.lang.String dn, java.lang.Class[] interfaces) Use to create a new LDAP entry
[illegible]	**remove**(java.lang.Object entry) Use to remove an existing entry from the Directory
[illegible]	**search**(java.lang.String ctxToSearch, java.lang.String filter) Use to execute a query against the Directory using the specified 'ctxToSearch', and 'filter'
java.util.List	**search**(java.lang.String ctxToSearch, java.lang.String filter, javax.naming.directory.SearchControls searchCtrls) Use to execute a query against the Directory using the specified 'ctxToSearch', 'filter', and 'searchCtrls.
java.util.List	**search**(java.lang.String ctxToSearch, java.lang.String filter, javax.naming.directory.SearchControls searchCtrls, javax.naming.ldap.Control[] reqCtrls) Use to execute a query against the Directory using the specified 'ctxToSearch', 'filter', 'searchCtrls, and 'reqCtrls'

	write(java.lang.Object entry) Use to commit a new entry or modifications of an existing entry to the Directory

Method Detail

newEntryInstance

```
public java.lang.Object newEntryInstance(java.lang.String dn,
                                         java.lang.Class[] interfaces)
                                  throws javax.naming.NamingException
```

Use to create a new LDAP entry The entry is not written to the Directory until DirectoryManager write() is executed.

Parameters:

`dn` - Distinquished name for the new entry Must not be null or empty

`interfaces` - Array of Class objects which represent the interfaces that this new entry will support The Class objects MUST be one of the LDAP code generated interfaces Array must not be null or empty

Returns:

Object representing the directory entry This object can be cast to the appropriate "objectclass" interface (s)

Throws:

`javax.naming.NamingException` - if a naming exception is encountered

---

mixinInterfaces

```
public java.lang.Object mixinInterfaces(java.lang.Object entry,
                                        java.lang.Class[] newInterfaces)
                                 throws javax.naming.NamingException
```

Use to add additional interfaces specified by 'newInterfaces' to an existing 'entry'. The modified entry is not written to the Directory until DirectoryManager.write() is executed.

Parameters:

`entry` - Existing LDAP entry to mix new interfaces into 'entry' must be acquired by calls to DirectoryController lookup(), DirectoryController search(), or DirectoryController newEntryInstance()

`newInterfaces` - Array of new interfaces to mix into the entry. Must not be null and must not be empty

Returns:

Object representing the modified directory entry This object can be cast to the appropriate "objectclass" interface(s) including those contained in 'newInterfaces'.

Throws:

`javax.naming.NamingException` - if a naming exception is encountered

---

lookup

```
public java.lang.Object lookup(java.lang.String dn)
                         throws javax.naming.NameNotFoundException,
                                javax.naming.NamingException,
                                java.lang.ClassNotFoundException
```

Use to lookup a specific entry identified by the specified 'dn'

Parameters:

dn - The distinquished name which uniquely identifies the entry Must not be null and must not be empty

Returns:

Object representing the directory entry bound to the specified dn This object can be cast to the appropriate "objectclass" interface(s)

Throws:

javax.naming.NameNotFoundException - if dn cannot be resolved because it is not bound

javax.naming.NamingException - if a naming exception is encountered

java.lang.ClassNotFoundException - if the looked up entry contains an object class which does not have an associated code generated interface

---

search

```
public java.util.List search(java.lang.String ctxToSearch,
                             java.lang.String filter)
                      throws javax.naming.NamingException,
                             java.lang.ClassNotFoundException
```

Use to execute a query against the Directory using the specified 'ctxToSearch', and 'filter'

Parameters:

ctxToSearch - Context to search "" for current context Must not be null

filter - LDAP search filter Must not be null

Returns:

List of Objects representing the results of the search These Object can each be cast to the appropriate "objectclass" interface(s). If search finds nothing, List returned will have size of zero Return will never be null

Throws:

java.lang.ClassNotFoundException - if the entries found contains an object class which does not have an associated code generated interface

javax.naming.NamingException - if naming exception is encountered.

---

search

```
public java.util.List search(java.lang.String ctxToSearch,
                             java.lang.String filter,
                             javax.naming.directory.SearchControls searchCtrls)
                      throws javax.naming.NamingException,
                             java.lang.ClassNotFoundException
```

Use to execute a query against the Directory using the specified 'ctxToSearch', 'filter', and 'searchCtrls

Parameters:

`ctxToSearch` - Context to search "" for current context. Must not be null

`filter` - LDAP search filter Must not be null

`searchCtrls` - Used to determine scope of search and what gets returned May be null If null, defaults will be used (search using SearchControls SUBTREE_SCOPE)

Returns:

List of Objects representing the results of the search These Object can each be cast to the appropriate "objectclass" interface(s) If search finds nothing, List returned will have size of zero Return will never be null

Throws:

`java.lang.ClassNotFoundException` - if the entries found contains an object class which does not have an associated code generated interface

`javax.naming.NamingException` - if naming exception is encountered.

See Also:

`SearchControls`

---

search

```
public java.util.List search(java.lang.String ctxToSearch,
                             java.lang.String filter,
                             javax.naming.directory.SearchControls searchCtrls,
                             javax.naming.ldap.Control[] reqCtrls)
                      throws javax.naming.NamingException,
                             java.lang.ClassNotFoundException
```

Use to execute a query against the Directory using the specified 'ctxToSearch', 'filter', 'searchCtrls, and 'reqCtrls'

Parameters:

`ctxToSearch` - Context to search "" for current context Must not be null.

`filter` - LDAP search filter. Must not be null

`searchCtrls` - Used to determine scope of search and what gets returned May be null If null, defaults will be used (search using SearchControls.SUBTREE_SCOPE).

`reqCtrls` - A control to request the LDAP search to return in a certain way (i e, sort results in a particular way) May be null. If null, no LDAP request controls will be used

Returns:

List of Objects representing the results of the search These Object can each be cast to the appropriate "objectclass" interface(s). If search finds nothing, List returned will have size of zero. Return will never be null

Throws:

`java.lang.ClassNotFoundException` - if the entries found contains an object class which does not have an associated code generated interface.

`javax.naming.NamingException` - if naming exception is encountered

See Also:

`SearchControls, Control`

---

write

```
public void write(java.lang.Object entry)
          throws javax.naming.NamingException
```

Use to commit a new entry or modifications of an existing entry to the Directory

Parameters:

`entry` - Entry to commit to the directory 'entry' must have been acquired by calls to DirectoryController lookup(), DirectoryController search(), or DirectoryController newEntryInstance() Must not be null

Throws:

`javax.naming.NamingException` - if naming exception is encountered.

---

remove

```
public void remove(java.lang.Object entry)
            throws javax.naming.NamingException
```

Use to remove an existing entry from the Directory.

Parameters:

`entry` - Entry to remove from the directory. 'entry' must have been acquired by calls to DirectoryController lookup(), DirectoryController search(), or DirectoryController newEntryInstance() Must not be null

Throws:

`javax.naming.NamingException` - if naming exception is encountered

---

cast

```
public java.lang.Object cast(java.lang.Object entry,
                             java.lang.Class interfaceToCastTo)
                      throws java.lang.ClassCastException
```

This method provides backward compatibility from Java 1.3 to Java 1.2 which does not support the Proxy class This method is not yet implemented

Parameters:

`entry` - Entry to cast. 'entry' must have been acquired by calls to DirectoryController.lookup(), DirectoryController search(), or DirectoryController.newEntryInstance() Must not be null

`interfaceToCastTo` - This is the interface that the specfied 'entry' is to be cast to

Returns:

Object which can be cast to the type specified by 'interfaceToCastTo'.

Throws:

`java.lang.ClassCastException` - if the specified 'entry' cannot be cast to the specified

'interfaceToCastTo'

---

getDN

```
public java.lang.String getDN(java.lang.Object entry)
```

Use to get the specified 'entry' distinquishe ' ıame

Parameters:

`entry` - Entry to obtain distinquished name from 'entry' must have been acquired by calls to DirectoryController lookup(), DirectoryController search(), or DirectoryController new EntryInstance( ) Must not be null.

Returns:

String containing the specified 'entry' distinquished name

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL FIELD | CONSTR | METHOD

com.gepower.sfo.tool.ldap

Class DefaultDirectorySource

```
java.lang.Object
  |
  +--com.gepower.sfo.tool.ldap.DefaultDirectorySource
```

All Implemented Interfaces:
DirectorySource

---

public class **DefaultDirectorySource**
extends java.lang Object
implements DirectorySource

DirectorySource implementation See DirectorySource for discription of implemented methods

---

Constructor Summary

DefaultDirectorySource(java.util.Hashtable environment)

Method Summary

void	**discardDirContext**(javax.naming.directory.DirContext context) Use to discard the specified 'context'.
javax.naming.directory.DirContext	**getDirContext**() Use to get a JNDI DirContext object
void	**releaseDirContext**(javax.naming.directory.DirContext context) Use to release the specified 'context'

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DefaultDirectorySource

```
public DefaultDirectorySource(java.util.Hashtable environment
                        throws javax.naming.NamingException
```

Method Detail

getDirContext

```
public javax.naming.directory.DirContext getDirContext()
                                                  throws javax.naming.NamingException
```

Description copied from interface: `DirectorySource`
Use to get a JNDI DirContext object
Specified by:
`getDirContext` in interface `DirectorySource`
Following copied from interface. `com.gepower.sfo.tool.ldap.DirectorySource`
Returns:
DirContext object
Throws:
`javax.naming.NamingException` - if a naming exception is encountered

---

releaseDirContext

```
public void releaseDirContext(javax.naming.directory.DirContext context)
```

Description copied from interface: `DirectorySource`
Use to release the specified 'context'. This should be called when the context is no longer needed.
Specified by:
`releaseDirContext` in interface `DirectorySource`
Following copied from interface `com.gepower.sfo.tool.ldap.DirectorySource`
Parameters:
`context` - The context to release.

---

discardDirContext

```
public void discardDirContext(javax.naming.directory.DirContext context)
```

Description copied from interface: `DirectorySource`
Use to discard the specified 'context'
Specified by:
`discardDirContext` in interface `DirectorySource`

Following copied from interface: `com.gepower.sfo.tool.ldap.DirectorySource`

Parameters:

`context` - The context to release

com.gepower.sfo.tool.ldap

Class DirectoryEntry

```
java.lang.Object
  |
  +--com.gepower.sfo.tool.ldap.DirectoryEntry
```

All Implemented Interfaces:

java.lang.reflect.InvocationHandler, java.io.Serializable

---

public class **DirectoryEntry**
extends java.lang.Object
implements java.io.Serializable, java.lang.reflect.InvocationHandler

Represents an LDAP Directory Entry and a LDAP invocation handler used in Proxy instances. Each proxy instance has an associated invocation handler. When a method is invoked on a proxy instance, the method invocation is encoded and dispatched to the invoke method of its invocation handler. This is a package scope class and not used directly by clients

See Also:

`InvocationHandler, java.lang.reflect.Proxy`, Serialized Form

Method Summary

java.lang.Object	**invoke**(java.lang.Object proxy, java.lang.reflect.Method method, java.lang.Object[] args) Implement abstract method invoke() from InvocationHandler
java.lang.String	**toString**(java.lang.Object proxy, java.lang.reflect.Method method) Returns the contents of all attribute in this entry

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Detail

invoke

```
public java.lang.Object invoke(java.lang.Object proxy,
                               java.lang.reflect.Method method,
                               java.lang.Object[] args)
                        throws java.lang.Throwable
```

Implement abstract method invoke() from InvocationHandler This method only recognizes methods that have been declared in the generated interfaces

Specified by:

`invoke` in interface `java.lang.reflect.InvocationHandler`

Parameters:

`proxy` - the proxy instance that the method was invoked on.

`method` - the Method instance corresponding to the interface method invoked on the proxy instance The declaring class of the Method object will be the interface that the method was declared in, which may be a superinterface of the proxy interface that the proxy class inherits the method through.

`args` - an array of objects containing the values of the arguments passed in the method invocation on the proxy instance, or null if interface method takes no arguments. Arguments of primitive types are wrapped in instances of the appropriate primitive wrapper class, such as java.lang.Integer or java.lang.Boolean.

Throws:

`java.lang.Throwable` - the exception to throw from the method invocation on the proxy instance The exception's type must be assignable either to any of the exception types declared in the throws clause of the interface method or to the unchecked exception types java.lang.RuntimeException or java.lang Error If a checked exception is thrown by this method that is not assignable to any of the exception types declared in the throws clause of the interface method, then an UndeclaredThrowableException containing the exception that was thrown by this method will be thrown by the method invocation on the proxy instance

See Also:

`java.lang.reflect.UndeclaredThrowableException`

---

toString

```
public java.lang.String toString(java.lang.Object proxy,
                                 java.lang.reflect.Method method)
```

Returns the contents of all attribute in this entry Use for debugging purposes only

Parameters:

`proxy` - The Proxy object serviced by this InvocationHandler.

`method` - The Method object invoked on the Proxy

Returns:

The contents of all attributes in this entry

---

com.gepower.sfo.tool.ldap

Class DirectoryManagerFactory

```
java.lang.Object
   |
   +--com.gepower.sfo.tool.ldap.DirectoryManagerFactory
```

public class **DirectoryManagerFactory**
extends java lang Object

Use to create an object which implements the DirectoryManager interface.

Constructor Summary

`DirectoryManagerFactory()`

Method Summary

static DirectoryManager	**newDirectoryManager**(DirectorySource src, java.lang.String pkg) Creates a new object which implements the DirectoryManager interface using the specified 'src' and 'pkg'
static DirectoryManager	**newDirectoryManager**(DirectorySource src, java.lang.String pkg, java.lang.ClassLoader loader) Creates a new object which implements the DirectoryManager interface using the specified 'src', 'pkg', and 'loader'.
static DirectoryManager	**newDirectoryManager**(DirectorySource src, java.lang.String pkg, java.lang.ClassLoader loader, java.io.PrintStream logger) Creates a new object which implements the DirectoryManager interface using the specified 'src', 'pkg', 'loader', and 'logger'
static DirectoryManager	**newDirectoryManager**(java.util.Hashtable env, java.lang.String pkg) Creates a new object which implements the DirectoryManager interface using the specified 'env' and 'pkg'.

static DirectoryManager	**newDirectoryManager**(java.util.Hashtable env, java.lang.String pkg, java.lang.ClassLoader loader) Creates a new object which implements the DirectoryManager interface using the specified 'env', 'pkg', and 'loader'
static DirectoryManager	**newDirectoryManager**(java.util.Hashtable env, java.lang.String pkg, java.lang.ClassLoader loader, java.io.PrintStream logger) Creates a new object which implements the DirectoryManager interface using the specified 'env', 'pkg', 'loader', and 'logger'

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

DirectoryManagerFactory

```
public DirectoryManagerFactory()
```

Method Detail

newDirectoryManager

```
public static DirectoryManager newDirectoryManager(java.util.Hashtable env,
                                                   java.lang.String pkg)
                                            throws java.lang.IllegalArgumentException,
                                                   javax.naming.NamingException
```

Creates a new object which implements the DirectoryManager interface using the specified 'env' and 'pkg'

Parameters:

`env` - Used to specify various preferences and properties that define the environment in which naming and directory services are accessed. Must not be null

`pkg` - The java package in which the LDAP interfaces were generated under Must not be null.

Throws:

`java.lang.IllegalArgumentException` - if 'env' or 'pkg' is null.

`javax.naming.NamingException` - if a naming exception is encountered

---

newDirectoryManager

```
public static DirectoryManager newDirectoryManager(java.util.Hashtable env,
                                                   java.lang.String pkg,
                                                   java.lang.ClassLoader loader)
                                            throws java.lang.IllegalArgumentException,
```

javax.naming.NamingException

Creates a new object which implements the DirectoryManager interface using the specified 'env', 'pkg', and 'loader'

Parameters:

env - Used to specify various preferences and properties that define the environment in which naming and directory services are accessed Must not be null.

pkg - The java package in which the LDAP interfaces were generated under Must not be null

loader - Class loader to use to load proxy classes May be null in which case the current threads class loader will be used

Throws:

java.lang.IllegalArgumentException - if 'env' or 'pkg' is null.

javax.naming.NamingException - if a naming exception is encountered.

---

newDirectoryManager

```
public static DirectoryManager newDirectoryManager(java.util.Hashtable env,
                                                    java.lang.String pkg,
                                                    java.lang.ClassLoader loader,
                                                    java.io.PrintStream logger)
                                             throws java.lang.IllegalArgumentException,
                                                    javax.naming.NamingException
```

Creates a new object which implements the DirectoryManager interface using the specified 'env', 'pkg', 'loader', and 'logger'

Parameters:

env - Used to specify various preferences and properties that define the environment in which naming and directory services are accessed. Must not be null.

pkg - The java package in which the LDAP interfaces were generated under Must not be null

loader - Class loader to use to load proxy classes May be null in which case the current threads class loader will be used.

logger - This is where all debug trace messages will be written to.

Throws:

java.lang.IllegalArgumentException - if 'env' or 'pkg' is null.

javax.naming.NamingException - if a naming exception is encountered

---

newDirectoryManager

```
public static DirectoryManager newDirectoryManager(DirectorySource src,
                                                    java.lang.String pkg)
                                             throws java.lang.IllegalArgumentException,
                                                    javax.naming.NamingException
```

Creates a new object which implements the DirectoryManager interface using the specified 'src' and 'pkg'

Parameters:

src - Specifies the what directory source the DirectoryManager will use. Must not be null.

pkg - The java package in which the LDAP interfaces were generated under Must not be null

Throws:

java.lang.IllegalArgumentException - if 'src' or 'pkg' is null

javax.naming.NamingException - if a naming exception is encountered.

---

newDirectoryManager

```
public static DirectoryManager newDirectoryManager(DirectorySource src,
                                                    java.lang.String pkg,
                                                    java.lang.ClassLoader loader,
                                                  throws java.lang.IllegalArgumentException,
                                                    javax.naming.NamingException
```

Creates a new object which implements the DirectoryManager interface using the specified 'src', 'pkg', and 'loader'.

Parameters:

src - Specifies the what directory source the DirectoryManager will use Must not be null

pkg - The java package in which the LDAP interfaces were generated under. Must not be null.

loader - Class loader to use to load proxy classes May be null in which case the current threads class loader will be used.

Throws:

java.lang.IllegalArgumentException - if 'src' or 'pkg' is null

javax.naming.NamingException - if a naming exception is encountered

---

newDirectoryManager

```
public static DirectoryManager newDirectoryManager(DirectorySource src,
                                                    java.lang.String pkg,
                                                    java.lang.ClassLoader loader,
                                                    java.io.PrintStream logger)
                                                  throws java.lang.IllegalArgumentException,
                                                    javax.naming.NamingException
```

Creates a new object which implements the DirectoryManager interface using the specified 'src', 'pkg', 'loader', and 'logger'

Parameters:

src - Specifies the what directory source the DirectoryManager will use. Must not be null.

pkg - The java package in which the LDAP interfaces were generated under. Must not be null.

loader - Class loader to use to load proxy classes. May be null in which case the current threads class loader will be used.

logger - This is where all debug trace messages will be written to.

Throws:

java.lang.IllegalArgumentException - if 'src' or 'pkg' is null

javax.naming.NamingException - if a naming exception is encountered.

---

com.gepower.sfo.tool.ldap

Class Generator

```
java.lang.Object
   |
   +--com.gepower.sfo.tool.ldap.Generator
```

public abstract class **Generator**
extends java.lang Object

Generates java interfaces which represents LDAP object classes. These classes are used in the java LDAP Directory framework. This class is abstract can contains only static methods. This class contains a main() method and is designed to be executed from the command line. See method description for main() for more details

See Also:
main(java.lang.String[])

Constructor Summary

Generator()

Method Summary

static void	main(java.lang.String[] args) Usage: java com.gepower.sfo.tool.ldap.Generator params [options]

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Generator

```
public Generator()
```

Method Detail

main

```
public static void main(java.lang.String[] args)
```

Usage java com.gepower.sfo tool.ldap Generator params [options]

To print out help, use the -help option when executing this program from the command line

Parameters:

`args` - Array of String arguments which consists of the required and optional parameters

Required Parameters

- '-sourcerootpath' the root directory path for the generated java source
- '-package' the java package for the generated java source
- '-dirctxfactory' class to use for the initial directory context factory
- '-providerurl' the LDAP URL string (i.e., ldap //localhost 389/o=ge com)
- '-securityprincipal' identity of the principal for authenticating the caller to the service
- '-securitycredentials' credentials of the principal for authenticating the caller to the service
- '-securityauthentication' security level to use

Optional Parameters:

- '-exclude' object classes matching the wildcard will be excluded from code generation Exclusions have precedence over Inclusion Multiple wildcards can be specified separated by semi-colons (i e. "ns*, ob*, net*server")
- '-include' object classes matching the wildcard will be included in code generation If option not specified, include all object classes Multiple wildcards can be specified, see exclude option
- '-version' version number that will be included into the javadoc of the generated code.
- '-tabstop' tab stop to use when formatting the generated code
- '-help' use to print usage syntax on the command line
- '-?' use to print usage syntax on the command line

Appendix D

Package com.ge.casper.http.jsp

Provides the interfaces and classes that define the contract that JspPreparer subclasses have with the framework.

See:

Description

Class Summary	
JspPreparer	Provides an abstract class to be subclassed for creating a view handler that prepares data beans and resources for subsequent use by a JSP that will render the response to the client.

Package com.ge.casper.http.jsp Description

Provides the interfaces and classes that define the contract that JspPreparer subclasses have with the framework.

com.ge.casper.http.jsp

Class JspPreparer

```
java.lang.Object
  |
  +--com.ge.casper.http.jsp.JspPreparer
```

All Implemented Interfaces:
ViewHandler

---

public abstract class **JspPreparer**
extends java.lang.Object
implements ViewHandler

Provides an abstract class to be subclassed for creating a view handler that prepares data beans and resources for subsequent use by a JSP that will render the response to the client.

A subclass must override the doService method, and may override the doInit and doDestroy methods.

Version:
1.0
Author:
Jeff Tuatini

Field Summary	
static java.lang.String	JSP_NAME

Constructor Summary
JspPreparer()

Method Summary	
void	**destroy**() Called by the framework to indicate to a translator that it is being taken out of service.
protected void	**doDestroy**() Overridden by the subclass to release resources.
protected void	**doInit**(ViewHandlerConfig config, HttpContainerContext ctx) Overridden by the subclass to perform initialization.
protected java.lang.String	**doPrepareForJsp**(ViewRequest vreq, HttpContainerRequestContext ctx) Overridden by the subclass to prepare data beans and resources from the view request for subsequent use by a JSP.
void	**init**(ViewHandlerConfig config) Called by the framework to indicate to a view handler that it is being placed into service.
void	**service**(ViewRequest vreq, ResponseChannel out, ContainerRequestContext containerCtx) Called by the framework to allow a view handler to transform and return a response back to the external client.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

JSP_NAME

```
public static final java.lang.String JSP_NAME
```

Constructor Detail

JspPreparer

```
public JspPreparer()
```

Method Detail

init

```
public final void init(ViewHandlerConfig config)
```

```
        throws SystemException
```

Description copied from interface: `ViewHandler`
Called by the framework to indicate to a view handler that it is being placed into service.

The framework calls the `init` method exactly once after instantiating the view handler. The `init` method must complete successfully before the view handler can receive any requests.

Specified by:
`init` in interface `ViewHandler`
Following copied from interface: `com.ge.casper.app.view.ViewHandler`
Parameters:
`config` - a `ViewHandlerConfig` object containing the view handler's configuration and initialization parameters. Also provides access to the `EnvironmentContext` object that enables access to the resources of the application.
Throws:
`SystemException` - if an exception has occurred that interferes with the view handler's normal operation.

---

destroy

```
public final void destroy()
```

Description copied from interface: `ViewHandler`
Called by the framework to indicate to a translator that it is being taken out of service.
Specified by:
`destroy` in interface `ViewHandler`

---

service

```
public final void service(ViewRequest vreq,
                          ResponseChannel out,
                          ContainerRequestContext containerCtx)
                   throws SystemException
```

Description copied from interface: `ViewHandler`
Called by the framework to allow a view handler to transform and return a response back to the external client.

The response returned from the action handler that is to be transformed is contained in the `ViewRequest` argument. The `ResponseChannel` argument is used by the view handler to return the transformed response to the client.

This method is only called after the view handler's `init` method has completed successfully.

View handlers run inside a multithreaded environment in which multiple responses must be transformed concurrently. Access to the view handler's class and instance variable must

therefore be synchronized if they are updateable within the `service` method.

Specified by:

`service` in interface `ViewHandler`

Following copied from interface: `com.ge.casper.app.view.ViewHandler`

Parameters:

`req` - a `ViewRequest` object that contains the response message to be transformed.

`out` - a `ResponseChannel` object that the view handler will use to return the transformed response to the client.

`context` - the `ContainerRequestContext` used to provide access to container adapter contextual information and resources related to the request. Container adapters may provide specialized subinterfaces to provide access to resources specific to the container adapter type.

Throws:

`SystemException` - if an exception has occurred that interferes with the view handler's normal operation.

---

doInit

```
protected void doInit(ViewHandlerConfig config,
                      HttpContainerContext ctx)
               throws SystemException
```

Overridden by the subclass to perform initialization.

---

doPrepareForJsp

```
protected java.lang.String doPrepareForJsp(ViewRequest vreq,
                                           HttpContainerRequestContext ctx)
                                    throws SystemException
```

Overridden by the subclass to prepare data beans and resources from the view request for subsequent use by a JSP. Generally, the data beans and resources that are prepared will be made available to a subsequent JSP as named attributes of a servlet object (eg, ServletRequest) accessible from the `HttpContainerRequestContext` and `HttpContainerContext` objects.

Upon the subclass returning from this method call, this base class will dispatch a JSP to render the response to the client. The JSP that is dispatched is determined by the return value of this method call. If this method returns `null`, the JSP whose name is specified with the JSP initialization parameter in the `casper-application.xml` file is dispatched. If this method returns a string value, the string specifies the name of the JSP to be dispatched.

---

doDestroy

```
protected void doDestroy()
```

Overridden by the subclass to release resources.

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

com.ge.casper.app.view

Interface ViewHandler

All Known Implementing Classes:
HttpViewHandler, JspPreparer

---

public interface **ViewHandler**

Defines methods that all view handler components must implement.

A view handler component is responsible for transforming the response returned from an action handler into a presentation supported by the client. The application framework selects a view handler to perform this task based upon the logical view name returned from the action handler and the response encoding supported by the client.

There is a wide range of possible implementations and capabilities for view handlers; a view handler instance may be specialized for a specific logical view name and response encoding, or it may be generic supporting all logical view names and response encodings.

ViewHandler Declaration

View handlers are declared in `casper-application.xml` configuration files with `view-handler` elements. The mapping of view handlers to views are specified with `view-handler-mapping` elements which are grouped by encoding. Refer to the `casper-application-1.0.dtd` for more details on configuration. The framework reads these configuration files in order of narrowing scope, and reads the elements within each configuration file in the order that they are declared. A `view-handler` element will replace an earlier declared element of the same type and name. A `view-handler-mapping` element will replace an earlier declared `view-handler-mapping` element of the same name under the same encoding.

A single view handler may be mapped to multiple views across different encodings.

The following encoding and view names have special meaning to the framework. How these names are used in the algorithm that the framework uses to retrieve a view handler for returning a response is described later.

- A view handler mapped to a view declared for the `ANY` encoding supports the view for any encoding.
- A view handler mapped to an `ANY` view supports any view under the encoding that the `ANY` view is declared. There may only be a single `ANY` view for each encoding.
- A view handler mapped to a `SYSTEM-ERROR` view is called by the framework to return an error message under the encoding that the `SYSTEM-ERROR` view is declared. There must be declared a

SYSTEM-ERROR view for each encoding.

ViewHandler Selection

The application framework follows the following algorithm to select an view handler to return a response message:

- The application framework retrieves a view handler supporting the given view name and encoding.
- If none exists, the framework will retrieve the view handler that is configured for the ANY view and the given encoding.
- If none exists, the framework will retrieve the view handler that is configured for the given view name and the ANY encoding.
- If none exists, the framework will retrieve the view handler that is configured for the ANY view and the ANY encoding.
- If none exists, the framework will display an error message using the view handler for the SYSTEM-ERROR view and given encoding. There must be a view handler for the SYSTEM-ERROR view for every encoding.

ViewHandler Lifecycle

This interface defines methods to initialize a view handler, to transform responses, and to remove a view handler from the framework. These methods are called in the following sequence:

- The view handler is constructed, then initialized with the `init` method.
- Any calls from the framework to transform and return responses are handled.
- The view handler is taken out of service, then destroyed with the `destroy` method, then garbage collected and finalized.

ViewHandler Concurrency

Within an application, there is only a single instance of each view handler. An instance of a view handler may be executed concurrently in multiple threads to transform multiple responses for returning to multiple clients. Therefore, a view handler must be programmed to be thread safe.

Version:
1.0
Author:
Jeff Tuatini

Method Summary	
void	**destroy**() Called by the framework to indicate to a translator that it is being taken out of service.
void	**init**(ViewHandlerConfig config) Called by the framework to indicate to a view handler that it is being placed into service.

void	**service**(ViewRequest req, ResponseChannel out, ContainerRequestContext context) Called by the framework to allow a view handler to transform and return a response back to the external client.

Method Detail

init

```
public void init(ViewHandlerConfig config)
          throws SystemException
```

Called by the framework to indicate to a view handler that it is being placed into service.

The framework calls the `init` method exactly once after instantiating the view handler. The `init` method must complete successfully before the view handler can receive any requests.

Parameters:

`config` - a `ViewHandlerConfig` object containing the view handler's configuration and initialization parameters. Also provides access to the `EnvironmentContext` object that enables access to the resources of the application.

Throws:

`SystemException` - if an exception has occurred that interferes with the view handler's normal operation.

---

service

```
public void service(ViewRequest req,
                    ResponseChannel out,
                    ContainerRequestContext context)
             throws SystemException
```

Called by the framework to allow a view handler to transform and return a response back to the external client.

The response returned from the action handler that is to be transformed is contained in the `ViewRequest` argument. The `ResponseChannel` argument is used by the view handler to return the transformed response to the client.

This method is only called after the view handler's `init` method has completed successfully.

View handlers run inside a multithreaded environment in which multiple responses must be transformed concurrently. Access to the view handler's class and instance variable must therefore be synchronized if they are updateable within the `service` method.

Parameters:

`req` - a `ViewRequest` object that contains the response message to be transformed.
`out` - a `ResponseChannel` object that the view handler will use to return the transformed response to the client.
`context` - the `ContainerRequestContext` used to provide access to container adapter contextual information and resources related to the request. Container adapters may provide specialized subinterfaces to provide access to resources specific to the container adapter type.

Throws:

`SystemException` - if an exception has occurred that interferes with the view handler's normal operation.

---

destroy

```
public void destroy()
```

Called by the framework to indicate to a translator that it is being taken out of service.

Package com.ge.casper.http

Provides the interfaces and classes that defines the contract that view components have with a "http-servlet" container type in which the application is deployed as a Servlet web application.

See:
Description

Interface Summary	
HttpContainerContext	Extends the `ContainerContext` interface to provide access to the `ServletContext` of the http-servlet container.
HttpContainerRequestContext	Extends the `ContainerRequestContext` interface to define methods to access HTTP servlet resources maintained by the http-servlet container for the current request.
HttpLinkEncoder	Defines methods for encoding an action name and parameters into a URL.

Class Summary	
HttpJspDispatcherHandler	Provides an abstract class to be subclassed to assist in creating a view filter that requires access to a http-servlet specific resources.
HttpServletBridgeFilter	Provides an abstract class to be subclassed to assist in creating a view filter that requires access to a http-servlet specific resources.
HttpServletBridgeSingleton	
HttpViewFilter	Provides an abstract class to be subclassed to assist in creating a view filter that requires access to a http-servlet specific resources.
HttpViewHandler	Provides an abstract class to be subclassed to assist in creating a view filter that requires access to a http-servlet specific resources.

Package com.ge.casper.http Description

Provides the interfaces and classes that defines the contract that view components have with a "http-servlet" container type in which the application is deployed as a Servlet web application.

com.ge.casper.http

Interface HttpContainerContext

All Superinterfaces:
ContainerContext

public interface **HttpContainerContext**
extends ContainerContext

Extends the `ContainerContext` interface to provide access to the `ServletContext` of the http-servlet container.

Version:
1.0
Author:
Jeff Tuatini

Method Summary

`javax.servlet.ServletContext`	**`getServletContext`**`()` Returns the `ServletContext` object.

Methods inherited from interface com.ge.casper.app.container.ContainerContext
`getAdapterName`, `getAdapterVersion`, `getContainerType`

Method Detail

getServletContext

`public javax.servlet.ServletContext` **`getServletContext`**`()`

Returns the `ServletContext` object.

com.ge.casper.http

Interface HttpContainerRequestContext

All Superinterfaces:
ContainerRequestContext

public interface **HttpContainerRequestContext**
extends ContainerRequestContext

Extends the `ContainerRequestContext` interface to define methods to access HTTP servlet resources maintained by the http-servlet container for the current request.

Version:
1.0
Author:
Jeff Tuatini

Method Summary

HttpLinkEncoder	**getHttpLinkEncoder**() Returns the `HttpLinkEncoder` for encoding actions and URLs within the context of the current request.
javax.servlet.http.HttpServletRequest	**getHttpServletRequest**() Returns the `HttpServletRequest` underlying the current request.
javax.servlet.http.HttpServletResponse	**getHttpServletResponse**() Returns the `HttpServletResponse` underlying the current request.

Methods inherited from interface com.ge.casper.app.container.ContainerRequestContext
getSessionId, isUserInRole, logoutUser

Method Detail

getHttpLinkEncoder

```
public HttpLinkEncoder getHttpLinkEncoder()
```

Returns the `HttpLinkEncoder` for encoding actions and URLs within the context of the current request.

---

getHttpServletRequest

```
public javax.servlet.http.HttpServletRequest getHttpServletRequest()
```

Returns the `HttpServletRequest` underlying the current request.

---

getHttpServletResponse

```
public javax.servlet.http.HttpServletResponse getHttpServletResponse()
```

Returns the `HttpServletResponse` underlying the current request.

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

com.ge.casper.http

Interface HttpLinkEncoder

All Superinterfaces:
LinkEncoder

public interface **HttpLinkEncoder**
extends LinkEncoder

Defines methods for encoding an action name and parameters into a URL.

Version:
1.0
Author:
Jeff Tuatini

Method Summary

java.lang.String	**encodeAction**(java.lang.String action) Encodes the given action name into a URL.
java.lang.String	**encodeAction**(java.lang.String action, java.lang.String queryString) Encodes the given action name and query string into a URL.
java.lang.String	**encodeAction**(java.lang.String action, java.lang.String[] [] nvPairs) Encodes the given action name and parameters into a URL.
java.lang.String	**encodeRedirectAction**(java.lang.String action) Encodes the given action name into a URL for use in a sendRedirect call.
java.lang.String	**encodeRedirectAction**(java.lang.String action, java.lang.String queryString) Encodes the given action name and query string into a URL for use in a sendRedirect call.
java.lang.String	**encodeRedirectAction**(java.lang.String action, java.lang.String[] [] nvPairs) Encodes the given action name and parameters into a URL for use in a sendRedirect call.

java.lang.String	**encodeRedirectURL**(java.lang.String url) Encodes the given URL with any necessary container session data for use in a sendRedirect call.
java.lang.String	**encodeURL**(java.lang.String url) Encodes the given URL with any necessary container session data.

Method Detail

encodeAction

```
public java.lang.String encodeAction(java.lang.String action)
```

Encodes the given action name into a URL.

Specified by:

encodeAction in interface LinkEncoder

---

encodeAction

```
public java.lang.String encodeAction(java.lang.String action,
                                     java.lang.String[][] nvPairs)
```

Encodes the given action name and parameters into a URL.

Specified by:

encodeAction in interface LinkEncoder

---

encodeAction

```
public java.lang.String encodeAction(java.lang.String action,
                                     java.lang.String queryString)
```

Encodes the given action name and query string into a URL.

Specified by:

encodeAction in interface LinkEncoder

---

encodeURL

```
public java.lang.String encodeURL(java.lang.String url)
```

Encodes the given URL with any necessary container session data.

Specified by:

encodeURL in interface LinkEncoder

---

encodeRedirectAction

```
public java.lang.String encodeRedirectAction(java.lang.String action)
```

Encodes the given action name into a URL for use in a sendRedirect call.

---

encodeRedirectAction

```
public java.lang.String encodeRedirectAction(java.lang.String action,
                                             java.lang.String[][] nvPairs)
```

Encodes the given action name and parameters into a URL for use in a sendRedirect call.

---

encodeRedirectAction

```
public java.lang.String encodeRedirectAction(java.lang.String action,
                                             java.lang.String queryString)
```

Encodes the given action name and query string into a URL for use in a sendRedirect call.

---

encodeRedirectURL

```
public java.lang.String encodeRedirectURL(java.lang.String url)
```

Encodes the given URL with any necessary container session data for use in a sendRedirect call.

---

com.ge.casper.http

Class HttpJspDispatcherHandler

```
java.lang.Object
  |
  +--com.ge.casper.http.HttpViewHandler
        |
        +--com.ge.casper.http.HttpJspDispatcherHandler
```

All Implemented Interfaces:

ViewHandler

public class **HttpJspDispatcherHandler**
extends HttpViewHandler

Provides an abstract class to be subclassed to assist in creating a view filter that requires access to a http-servlet specific resources.

A subclass must override the doService method, and may override the doInit and doDestroy methods. The other protected methods are helper methods for use by the subclass to assist in providing storage and access to initialization patameters and context objects. This class also downcasts the container context objects to their HTTP subtypes as provided by the http-servlet container.

Developers need not extend this class to implement a ViewFilter that executes in a http-servlet container; they can implement the ViewFilter interface directly and perform the necessary downcasts to access HTTP specific resources.

Version:

1.0

Author:

Jeff Tuatini

Field Summary

static java.lang.String	**JSP_NAME**

Constructor Summary

HttpJspDispatcherHandler()

Method Summary

protected void	**doInit**() The subclass overrides this method to initialize itself.
protected void	**doService**(ViewRequest vreq, ResponseChannel out, HttpContainerRequestContext ctx) The subclass must override this method to process the view request.

Methods inherited from class com.ge.casper.http.HttpViewHandler

destroy, doDestroy, getEnvironmentContext, getHttpContainerContext, getInitParameter, getInitParameterNames, getName, getViewContext, init, service

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

JSP_NAME

```
public static final java.lang.String JSP_NAME
```

Constructor Detail

HttpJspDispatcherHandler

```
public HttpJspDispatcherHandler()
```

Method Detail

doInit

```
protected void doInit()
              throws SystemException
```

Description copied from class: HttpViewHandler
The subclass overrides this method to initialize itself.
Overrides:

doInit in class HttpViewHandler

---

doService

```
protected void doService(ViewRequest vreq,
                         ResponseChannel out,
                         HttpContainerRequestContext ctx)
                  throws SystemException
```

Description copied from class: HttpViewHandler
The subclass must override this method to process the view request. The chain parameter provides the interface for the subclass to invoke the next filter in the filter chain.
Overrides:
doService in class HttpViewHandler

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

com.ge.casper.http

Class HttpServletBridgeFilter

```
java.lang.Object
  |
  +--com.ge.casper.http.HttpViewFilter
        |
        +--com.ge.casper.http.HttpServletBridgeFilter
```

All Implemented Interfaces:
ViewFilter

---

public class **HttpServletBridgeFilter**
extends HttpViewFilter

Provides an abstract class to be subclassed to assist in creating a view filter that requires access to a http-servlet specific resources.

A subclass must override the doService method, and may override the doInit and doDestroy methods. The other protected methods are helper methods for use by the subclass to assist in providing storage and access to initialization patameters and context objects. This class also downcasts the container context objects to their HTTP subtypes as provided by the http-servlet container.

Developers need not extend this class to implement a ViewFilter that executes in a http-servlet container; they can implement the ViewFilter interface directly and perform the necessary downcasts to access HTTP specific resources.

Version:
1.0
Author:
Jeff Tuatini

Field Summary

static java.lang.String	**ENCODER**
static java.lang.String	**REQUEST**
static java.lang.String	**SESSION**

Constructor Summary

HttpServletBridgeFilter()

Method Summary

protected void	**doService**(ViewRequest req, ResponseChannel out, HttpContainerRequestContext c, ViewFilterChain chain) The subclass must override this method to process the view request.

Methods inherited from class com.ge.casper.http.HttpViewFilter

destroy, doDestroy, doInit, getEnvironmentContext, getHttpContainerContext, getInitParameter, getInitParameterNames, getName, getViewContext, init, service

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

SESSION

public static final java.lang.String **SESSION**

REQUEST

public static final java.lang.String **REQUEST**

ENCODER

public static final java.lang.String **ENCODER**

Constructor Detail

HttpServletBridgeFilter

public **HttpServletBridgeFilter**()

Method Detail

doService

```
protected void doService(ViewRequest req,
                         ResponseChannel out,
                         HttpContainerRequestContext c,
                         ViewFilterChain chain)
                  throws SystemException
```

Description copied from class: `HttpViewFilter`
The subclass must override this method to process the view request. The chain parameter provides the interface for the subclass to invoke the next filter in the filter chain.
Overrides:
`doService` in class `HttpViewFilter`

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

com.ge.casper.http

Class HttpServletBridgeSingleton

```
java.lang.Object
  |
  +--com.ge.casper.http.HttpServletBridgeSingleton
```

All Implemented Interfaces:

java.io.Serializable, Singleton

---

public class **HttpServletBridgeSingleton**
extends java.lang.Object
implements Singleton, java.io.Serializable

See Also:

Serialized Form

Field Summary	
static java.lang.String	**APP_CONTEXT**
static java.lang.String	**ENV_CONTEXT**
static java.lang.String	**LOGGER**
static java.lang.String	**VIEW_CONTEXT**

Constructor Summary
HttpServletBridgeSingleton()

Method Summary

void	**destroy**() Called by the framework to indicate to a singleton that it is being taken out of service.
void	**init**(SingletonConfig config) Called by the framework to indicate to a singleton that it is being placed into service.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

VIEW_CONTEXT

```
public static final java.lang.String VIEW_CONTEXT
```

APP_CONTEXT

```
public static final java.lang.String APP_CONTEXT
```

ENV_CONTEXT

```
public static final java.lang.String ENV_CONTEXT
```

LOGGER

```
public static final java.lang.String LOGGER
```

Constructor Detail

HttpServletBridgeSingleton

```
public HttpServletBridgeSingleton()
```

Method Detail

init

```
public void init(SingletonConfig config)
```

```
throws SystemException
```

Description copied from interface: `Singleton`
Called by the framework to indicate to a singleton that it is being placed into service.

The framework calls the `init` method exactly once after instantiating the singleton. The `init` method must complete successfully for the application to start. This method is called before any non-singleton components of the component package have been initialized.

Specified by:
`init` in interface `Singleton`

Following copied from interface: `com.ge.casper.app.Singleton`

Parameters:
`config` - a `SingletonConfig` object containing the singleton's configuration and initialization parameters. Also provides a reference to the `EnvironmentContext` object that enables access to CASPER services, and provides a reference to the component package `Context` subtype.

Throws:
`SystemException` - if an exception has occurred that interferes with the singleton's normal operation.

---

destroy

```
public void destroy()
```

Description copied from interface: `Singleton`
Called by the framework to indicate to a singleton that it is being taken out of service. This method is called after all non-singleton components of the component package have been destroyed.

Specified by:
`destroy` in interface `Singleton`

Overview Package Class Tree Deprecated Index Help
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

com.ge.casper.http

Class HttpViewFilter

```
java.lang.Object
  |
  +--com.ge.casper.http.HttpViewFilter
```

All Implemented Interfaces:
ViewFilter

Direct Known Subclasses:
HttpServletBridgeFilter

---

public abstract class **HttpViewFilter**
extends java.lang.Object
implements ViewFilter

Provides an abstract class to be subclassed to assist in creating a view filter that requires access to a http-servlet specific resources.

A subclass must override the doService method, and may override the doInit and doDestroy methods. The other protected methods are helper methods for use by the subclass to assist in providing storage and access to initialization patameters and context objects. This class also downcasts the container context objects to their HTTP subtypes as provided by the http-servlet container.

Developers need not extend this class to implement a ViewFilter that executes in a http-servlet container; they can implement the ViewFilter interface directly and perform the necessary downcasts to access HTTP specific resources.

Version:
1.0
Author:
Jeff Tuatini

Constructor Summary

HttpViewFilter()

Method Summary

void	**destroy**() Called by the framework to indicate to an view filter that the filter is being taken out of service.
protected void	**doDestroy**() The subclass overrides this method to destroy itself.
protected void	**doInit**() The subclass overrides this method to initialize itself.
protected abstract void	**doService**(ViewRequest req, ResponseChannel out, HttpContainerRequestContext ctx, ViewFilterChain chain) The subclass must override this method to process the view request.
protected EnvironmentContext	**getEnvironmentContext**() Returns a reference to the EnvironmentContext that provides the filter with access to the services of the application.
protected HttpContainerContext	**getHttpContainerContext**() Returns a reference to the HttpContainerContext object.
protected java.lang.String	**getInitParameter**(java.lang.String name) Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist
protected java.util.Iterator	**getInitParameterNames**() Returns the names of the view filter's initialization parameters as an Iterator of String objects, or an empty Iterator if the view filter has no initialization parameters
protected java.lang.String	**getName**() Returns the name of this filter instance.
protected ViewContext	**getViewContext**() Returns a reference to the ViewContext object.
void	**init**(ViewFilterConfig config) Called by the framework to indicate to an view filter that it is being placed into service.
void	**service**(ViewRequest req, ResponseChannel out, ContainerRequestContext ctx, ViewFilterChain chain) Called by the framework to invoke a filter with a request/response pair.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

HttpViewFilter

```
public HttpViewFilter()
```

Method Detail

init

```
public final void init(ViewFilterConfig config)
                throws SystemException
```

Description copied from interface: ViewFilter
Called by the framework to indicate to an view filter that it is being placed into service.

The framework calls the `init` method exactly once after instantiating the view filter. The `init` method must complete successfully before the view filter can receive any requests.

Specified by:
`init` in interface `ViewFilter`
Following copied from interface: `com.ge.casper.app.view.ViewFilter`
Parameters:
`config` - an `ViewFilterConfig` object containing the view filters's configuration and initialization parameters. Also provides a reference to the `EnvironmentContext` object that enables access to application services, and a reference to the `ViewContext` object.
Throws:
`SystemException` - if an exception has occurred that interferes with the filter's normal operation.

---

service

```
public final void service(ViewRequest req,
                          ResponseChannel out,
                          ContainerRequestContext ctx,
                          ViewFilterChain chain)
                   throws SystemException
```

Description copied from interface: ViewFilter
Called by the framework to invoke a filter with a request/response pair. An `ViewFilterChain` object is passed in this method to allow the filter to pass the request/response pair onto the next filter in the chain.
Specified by:
`service` in interface `ViewFilter`
Following copied from interface: `com.ge.casper.app.view.ViewFilter`
Parameters:
`req` - a `ViewRequest` object containing the data to be transformed for returning to the client.
`out` - a `ResponseChannel` object that is to be used to return the transformed response to the client.
`context` - the `ContainerRequestContext` used to provide access to container adapter

contextual information and resources related to the request. Container adapters may provide specialized subinterfaces to provide access to resources specific to the container adapter type.

`chain` - a `ViewFilterChain` object that is used to invoke the next filter in the chain.

Throws:

`SystemException` - if an exception has occurred that interferes with the filters's normal operation.

---

destroy

```
public final void destroy()
```

Description copied from interface: `ViewFilter`

Called by the framework to indicate to an view filter that the filter is being taken out of service.

Specified by:

`destroy` in interface `ViewFilter`

---

getName

```
protected java.lang.String getName()
```

Returns the name of this filter instance.

---

getInitParameter

```
protected java.lang.String getInitParameter(java.lang.String name)
```

Returns a `String` containing the value of the named initialization parameter, or null if the parameter does not exist

---

getInitParameterNames

```
protected java.util.Iterator getInitParameterNames()
```

Returns the names of the view filter's initialization parameters as an `Iterator` of `String` objects, or an empty `Iterator` if the view filter has no initialization parameters

---

getEnvironmentContext

```
protected EnvironmentContext getEnvironmentContext()
```

Returns a reference to the `EnvironmentContext` that provides the filter with access to the

services of the application.

---

getViewContext

```
protected ViewContext getViewContext()
```

Returns a reference to the `ViewContext` object.

---

getHttpContainerContext

```
protected HttpContainerContext getHttpContainerContext()
```

Returns a reference to the `HttpContainerContext` object.

---

doInit

```
protected void doInit()
               throws SystemException
```

The subclass overrides this method to initialize itself.

---

doService

```
protected abstract void doService(ViewRequest req,
                                  ResponseChannel out,
                                  HttpContainerRequestContext ctx,
                                  ViewFilterChain chain)
                           throws SystemException
```

The subclass must override this method to process the view request. The chain parameter provides the interface for the subclass to invoke the next filter in the filter chain.

---

doDestroy

```
protected void doDestroy()
```

The subclass overrides this method to destroy itself.

---

com.ge.casper.http

Class HttpViewHandler

```
java.lang.Object
  |
  +--com.ge.casper.http.HttpViewHandler
```

All Implemented Interfaces:
ViewHandler

Direct Known Subclasses:
HttpJspDispatcherHandler

---

public abstract class **HttpViewHandler**
extends java.lang.Object
implements ViewHandler

Provides an abstract class to be subclassed to assist in creating a view filter that requires access to a http-servlet specific resources.

A subclass must override the doService method, and may override the doInit and doDestroy methods. The other protected methods are helper methods for use by the subclass to assist in providing storage and access to initialization patameters and context objects. This class also downcasts the container context objects to their HTTP subtypes as provided by the http-servlet container.

Developers need not extend this class to implement a ViewFilter that executes in a http-servlet container; they can implement the ViewFilter interface directly and perform the necessary downcasts to access HTTP specific resources.

Version:
1.0
Author:
Jeff Tuatini

Constructor Summary

HttpViewHandler()

Method Summary	
void	**destroy**() Called by the framework to indicate to a translator that it is being taken out of service.
protected void	**doDestroy**() The subclass overrides this method to destroy itself.
protected void	**doInit**() The subclass overrides this method to initialize itself.
protected abstract void	**doService**(ViewRequest req, ResponseChannel out, HttpContainerRequestContext ctx) The subclass must override this method to process the view request.
protected EnvironmentContext	**getEnvironmentContext**() Returns a reference to the EnvironmentContext that provides the filter with access to the services of the application.
protected HttpContainerContext	**getHttpContainerContext**() Returns a reference to the HttpContainerContext object.
protected java.lang.String	**getInitParameter**(java.lang.String name) Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist
protected java.util.Iterator	**getInitParameterNames**() Returns the names of the view filter's initialization parameters as an Iterator of String objects, or an empty Iterator if the view filter has no initialization parameters
protected java.lang.String	**getName**() Returns the name of this filter instance.
protected ViewContext	**getViewContext**() Returns a reference to the ViewContext object.
void	**init**(ViewHandlerConfig config) Called by the framework to indicate to a view handler that it is being placed into service.
void	**service**(ViewRequest req, ResponseChannel out, ContainerRequestContext ctx) Called by the framework to allow a view handler to transform and return a response back to the external client.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

HttpViewHandler

```
public HttpViewHandler()
```

Method Detail

init

```
public final void init(ViewHandlerConfig config)
                throws SystemException
```

Description copied from interface: ViewHandler
Called by the framework to indicate to a view handler that it is being placed into service.

The framework calls the `init` method exactly once after instantiating the view handler. The `init` method must complete successfully before the view handler can receive any requests.

Specified by:
`init` in interface `ViewHandler`
Following copied from interface: `com.ge.casper.app.view.ViewHandler`
Parameters:
`config` - a `ViewHandlerConfig` object containing the view handler's configuration and initialization parameters. Also provides access to the `EnvironmentContext` object that enables access to the resources of the application.
Throws:
`SystemException` - if an exception has occurred that interferes with the view handler's normal operation.

---

service

```
public final void service(ViewRequest req,
                          ResponseChannel out,
                          ContainerRequestContext ctx)
                   throws SystemException
```

Description copied from interface: ViewHandler
Called by the framework to allow a view handler to transform and return a response back to the external client.

The response returned from the action handler that is to be transformed is contained in the `ViewRequest` argument. The `ResponseChannel` argument is used by the view handler to return the transformed response to the client.

This method is only called after the view handler's `init` method has completed successfully.

View handlers run inside a multithreaded environment in which multiple responses must be

transformed concurrently. Access to the view handler's class and instance variable must therefore be synchronized if they are updateable within the `service` method.

Specified by:

`service` in interface `ViewHandler`

Following copied from interface: `com.ge.casper.app.view.ViewHandler`

Parameters:

`req` - a `ViewRequest` object that contains the response message to be transformed.

`out` - a `ResponseChannel` object that the view handler will use to return the transformed response to the client.

`context` - the `ContainerRequestContext` used to provide access to container adapter contextual information and resources related to the request. Container adapters may provide specialized subinterfaces to provide access to resources specific to the container adapter type.

Throws:

`SystemException` - if an exception has occurred that interferes with the view handler's normal operation.

---

destroy

```
public final void destroy()
```

Description copied from interface: `ViewHandler`

Called by the framework to indicate to a translator that it is being taken out of service.

Specified by:

`destroy` in interface `ViewHandler`

---

getName

```
protected java.lang.String getName()
```

Returns the name of this filter instance.

---

getInitParameter

```
protected java.lang.String getInitParameter(java.lang.String name)
```

Returns a `String` containing the value of the named initialization parameter, or null if the parameter does not exist

---

getInitParameterNames

```
protected java.util.Iterator getInitParameterNames()
```

Returns the names of the view filter's initialization parameters as an `Iterator` of `String` objects, or an empty `Iterator` if the view filter has no initialization parameters

---

getEnvironmentContext

```
protected EnvironmentContext getEnvironmentContext()
```

Returns a reference to the `EnvironmentContext` that provides the filter with access to the services of the application.

---

getViewContext

```
protected ViewContext getViewContext()
```

Returns a reference to the `ViewContext` object.

---

getHttpContainerContext

```
protected HttpContainerContext getHttpContainerContext()
```

Returns a reference to the `HttpContainerContext` object.

---

doInit

```
protected void doInit()
              throws SystemException
```

The subclass overrides this method to initialize itself.

---

doService

```
protected abstract void doService(ViewRequest req,
                                  ResponseChannel out,
                                  HttpContainerRequestContext ctx)
                          throws SystemException
```

The subclass must override this method to process the view request. The chain parameter provides the interface for the subclass to invoke the next filter in the filter chain.

---

doDestroy

```
protected void doDestroy()
```

The subclass overrides this method to destroy itself.

Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD